ESD-TR-73-51, Vol. II

COMPUTER SECURITY TECHNOLOGY PLANNING STUDY

James P. Anderson

October 1972

DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
HQ ELECTRONIC SYSTEMS DIVISION (AFSC)
L. G. Hanscom Field, Bedford, Massachusetts 01730

Approved for public release;
distribution unlimited.

(Prepared under Contract No. F19628-72-C-0198 by James P. Anderson & Co., Box 42, Fort Washington, Pa. 19034.)

COMPUTER SECURITY TECHNOLOGY PLANNING STUDY

VOLUME II

JAMES P. ANDERSON

OCTOBER 1972

FOREWORD

This is Volume II of a two-volume report of the work of the Computer Security Technology Planning Study Panel. This work was performed under contract F19628-72-C-0198 in support of project 6917. This volume presents details supporting the recommended development plan. In addition, several papers prepared as part of the panel's activities are reproduced in the appendices. Appendices I and II were prepared by J. P. Anderson; Appendix III by E. Nelson; Appendix IV by C. Weissman; Appendix V by B. Peters, Appendix VI by E. L. Glaser, and Appendix VII by S. Lipner.

REVIEW AND APPROVAL

This technical report has been reviewed and approved.

MELVIN B. EMMONS, Colonel, USAF
Director, Information Systems Technology
Deputy for Command and Management Systems

ABSTRACT

Details of a planning study for USAF computer security requirements are presented. An Advanced development and Engineering program to obtain an open-use, multilevel secure computing capability is described. Plans are also presented for the related developments of communications security products and the interim solution to present secure computing problems. Finally a Exploratory development plan complementary to the recommended Advanced and Engineering development plans is also included.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

SECTION I

INTRODUCTION AND BACKGROUND

1.1 Background

In recent years the Air Force has become increasingly aware of the problem of computer security. This problem has intruded upon virtually every aspect of USAF operations and administration. The problem arises from a combination of factors that includes: greater reliance on the computer as a data processing and decision making tool in sensitive functional areas; the need to realize economies by consolidating ADP resources thereby integrating or co-locating previously separate data processing operations; the emergence of complex resource sharing computer systems providing users with capabilities for sharing data and processes with other users; the extension of resource sharing concepts to networks of computers; and the slowly growing recognition of security inadequacies of currently available computer systems. Most of the efforts to date to provide computer security have been centered in environments where all persons coming in contact with the system share a common clearance and where the principal effort has been directed to providing procedural controls, especially those associated with external access to the computer systems and their files, and proper marking of information found in the system.

1.2 Specific Security Problems of the USAF

The major problems of the USAF stem from the fact that there is a growing requirement to provide shared use of computer systems containing information of different classification levels and need-to-know requirements in a user population not uniformly cleared or access-approved. This problem takes an extreme form in those several systems currently under development or projected for the near future where part, or the majority of the user population has no clearance requirement and where only a very small fraction of the information being processed and stored on the systems is classified. In a few of the systems examined (see Section II below) the kinds of actions the user population is able to take are limited by the nature of the application in such a way as to avoid or reduce the security problem. However, in other systems, particularly in general use systems such as those found in the USAF Data Services Center in the Pentagon, the users are permitted and encouraged to directly program the system for their applications. It is in this latter kind of use of computers that the weakness of the technical foundation of current systems is most acutely felt.

Another major problem is the fact that there are growing pressures to interlink separate but related computer systems into increasingly complex networks. The principal problem seen here is that the security dangers of such interlinking are masked by the apparently "safe" interaction directly between computer systems.

Other problem areas in addition to those noted above generally fall into the category of techniques and technology available but not implemented in a form suitable for

the application to present and projected Air Force computer systems. Typical of this category is the notion of an "office environment" secure terminal. The technology for producing such terminals is both easily available and well understood but has not been clearly developed heretofore as an integrated requirement for the Air Force.

1.3 On The Nature Of The Security Threat

With the advent of widespread availability and use of resource-sharing systems, has come the realization that with the benefits of resource-sharing come problems of security and privacy that had not been recognized in previous batch systems. The key factor that permitted safe handling of classified information in the past was the fact that the computers were oriented to serving a single user at a time. Because of this, it was possible to isolate individual runs and apply security measures commensurate with the type of data being handled.

By the mid-sixties, the research in resource-sharing computer systems that had been going on in many universities had reached a stage of development that permitted a number of manufacturers to offer resource-sharing systems as a product. These products have formed the basis for the extensive application of resource-sharing to many systems found throughout the world today.

The interactive resource-sharing systems also provide economical centralization of programs and especially data online to an application that permits them to be accessed upon demand from any terminal attached to the system. This factor, plus the nature of time-sharing itself which provides for two or more programs to be resident simultaneously in primary storage, erodes the separation principle that had been the keystone to security practice in the past. Further, it replaces manual, easily visible controls with reliance upon logical and intangible program controls to keep separate data and programs belonging to different users.

At first glance, the problems of providing privacy and security in resource-shared systems seem ridiculously simple. Since it is a generally accepted requirement that the executive (operating system) for resource-shared systems and other users must be protected from 'buggy' programs, it follows that any of the various time-shared systems are 'secure'. Unfortunately, this is not the case.

The essence of the multilevel security technical problem becomes clear when the fact that programs of users with different clearances and data of different classifications share primary storage simultaneously in resource-sharing systems that rely on an operating system program to maintain their separation. Furthermore, the situation is aggravated when the user of a resource-sharing system, to a greater or lesser degree, must program the system to accomplish his work. In this environment, it is necessary to prove that a given system is proof against attack (i. e., hostile penetration).

It is generally true that contemporary systems provide limited protection against <u>accidental</u> violation of their operating systems; it is equally true that virtually none of

them provide any protection against deliberate attempts to penetrate the nominal security controls provided. It is the possibility of deliberate penetration by a user that we call malicious threat. It is the malicious threat that has forced most present systems to operate in single-level mode, where through the clearance process, all the users are considered equally reliable eliminating by definition the concern for maliciousness.

The malicious user concept arises from the requirements for open use systems. Present day computer systems are largely closed use systems; that is, systems serving a homogeneously cleared user population. The major threat to these systems is that of external penetration. The external penetration threat is countered by using combinations of physical, procedural and communications security techniques. These techniques, some highly advanced, are the bulk of the present state-of-the-art in computer security. In effect, the defense against external penetration surrounds the system and its user community with a barrier that must be breached before the system can be compromised. By adopting a uniform clearance (to the highest level of information contained in the systems), the threat of internal penetration is eliminated by definition.

The technical issue of multilevel computer security is concerned with the concept of malicious threat. By this we recognize that the nature of shared use multilevel computer systems present to a malicious user a unique opportunity for attempting to subvert through programming the <u>mechanism</u> upon which security depends (i. e., the control of the computer vested in the operating system). This threat, coupled with the concentration of the application (data, control system, etc.) in one place (the computer system) makes computers a uniquely attractive target for malicious (hostile) action. Recognition of the implication of malicious threat is important to understanding the security limitations surrounding application of contemporary computer systems. The threat that a single user of a system operating as a hostile agent can simply modify an operating system to by-pass or suspend security controls, and the fact that the operating system controlling the computer application(s) is developed outside of USAF control, contribute strongly to the reluctance to certify (i. e., be convinced) that contemporary systems are secure or even can be secured.

The objectives of providing open use multilevel systems differentiate users' clearances, and reduce the external control on physical access correspondingly. For systems operating on information at two or more security classification levels, it is mandatory that the system have security controls that are often not considered absolutely mandatory in a single level system due to the presumption of equal trustworthiness of all individuals using the system. The results of the requirements investigation have shown clearly that single level operation of many USAF systems is not either operationally or economically feasible. Further, <u>none</u> of the systems examined were found to be without a requirement to support a general programming capability, although in some applications-oriented (transaction) systems this is limited to a realtively small fraction of the total user population. Even in these systems, unless the application is developed using cleared implementors, the application(s) are such that while the users do not directly program the system, there is still no <u>assurance</u> that a programmed 'trap door' has not been installed in the application to be activated by some unique

string of input characters presented by collaberating user. Even if the application is developed by cleared implementors there is then no assurance, on present systems, that a 'trap door' has not been installed in a portion of the software base supporting the application.

The essence of this concern is that there exists manifold opportunities for a determined adversary to accomplish his objectives.

There is little question that contemporary commercially available systems do not provide an adequate defense against malicious threat. Most of these systems are known to have serious design and implementation flaws that can be exploited by individuals with programming access to the system. As an instance of this, we note that the Honeywell 6000 Series operating system has a number of major flaws that would permit a user programmer to subvert the nominal security controls that exist in the system. The design and implementation flaws in most contemporary systems permit a penetrating programmer to seize unauthorized control of the system, and thus have access to any of the information on the system.

In summary, the security threat is the demonstrated inability of most contemporary computer systems to provide a sufficiently strong technical defense against a malicious user who is deliberately attempting to penetrate the system for hostile purposes. The primary technical problem to be solved is that of determining what constitutes an appropriate defense against malicious attack, and then developing hardware and software with the defensive mechanism(s) built in.

1.4 Previous and Related Work

Because the problem of information security in computer based systems became visible only with the development of and acceptance of resource sharing systems, there is no long history of previous work. In 1967 the Defense Science Board Task Force on Computer Security was convened. It was intended that this Task Force would analyze the problem and recommend a research and development program that would provide solutions to the extant problems of that time. During the course of that work it was discovered that the problem was not well understood and as a consequence the final report prepared by the Task Force contained less in the way of a recommended R & D program than had originally been thought possible. The report did, however, contain an extensive discussion of the scope of the problem as well as definitions of terminology that were sadly lacking at that time.

During the past several years a number of independent projects concerned with various aspects of computer security have been funded by various members of the Defense and Intelligence communities. In addition, a fairly major effort to provide security controls to a system that existed within a benign environment in the Intelligence community has taken place over the past several years. While these controls are of interest and provide a certain degree of implementation of security procedures, they did not address the question of providing technical security against malicious attack.

More recently the Advanced Research Projects Agency (ARPA) has funded work at Rand Corporation, Information Systems Institute (USC) and Livermore Research Laboratories to analyze the security adequacy of selected commercial operating systems and to develop methodologies of security assurance. These programs are too recent and have not been sufficiently developed to provide any assessment of this potential contribution to the solution of some of the problems perceived by the study panel. Finally, the problem of computer security achieved major recognition from IBM's recent announcement of their intention to spend 40 million dollars on the problem over the next five years. The details of their program are unknown, but appear initially to be directed to the enhancement of an IBM product, Resource Security System (RSS).

1.5 Scope of this Study

1.5.1 Statement of Work

The scope of this study, as defined in the Statement of Work is:

> "The Contractor shall develop a comprehensive plan for research and development leading to the satisfaction of requirements for multi-user open computer systems which process various levels of classified and unclassified information simultaneously through terminals in both secure and insecure areas."

By 'open systems', we mean two things both of which are major contributors to the principal unsolved security problem facing the Air Force. First, we mean by open use, systems where not all of the users are cleared for the highest level of classification of information being processed on such a system. In the extreme, some users may not possess any clearance at all. Second, we mean by open use those systems where the users program the system in machine (assembly) language or any of the common higher order languages such as JOVIAL, FORTRAN, or COBOL. Either of these definitions of 'open system' creates unacceptable security hazards in contemporary systems. They serve to focus on the primary fact that too little is known about how the technical controls in the operating systems work to defend the system against attack, and assure that under no circumstances will classified information be inadvertently made available to an unauthorized user.

The emphasis on 'multi-user open systems' is well placed as this is the most stringent security environment we know. In addition to providing a useful model of severe security operating requirements, it is representative of a growing trend of use of computers in the USAF and other government departments. Further, technical solutions to the 'open systems' problems can be applied to less stringent environments as well.

1.5.2 Study Tasks

Specific tasks called for within this scope included:

a. A study and analysis of the security penetration threats and techniques as well as the effectiveness of current technology in meeting these threats, and the extent of research and development required to improve the current computer security technology.

b. An analysis of the state-of-the-art relating to the multi-user computer security problem to develop and recommend a technical program leading to the development of techniques which will satisfy USAF requirements for multilevel, open computer systems.

c. Identification of specific technical areas for which detailed plans will be developed.

d. Integration of the individual plans into a final comprehensive technical plan recommending how to satisfy the requirements for multi-user, multilevel secure computer systems which include terminals in both secure and unsecure areas.

1.5.3 Makeup of the Panel

Because of the complex interrelationships between various aspects of the problem, and to insure that all relevant aspects of the problem were considered, a study panel, chaired by Professor Edward L. Glaser of Case Western Reserve University was convened. Other members of the panel included:

- Mr. James P. Anderson, Deputy Chairman
- Dr. Melvin Conway
- Mr. Daniel J. Edwards (NSA)
- Miss Hilda Faust (NSA)
- Mr. Steven Lipner (MITRE) (Chairman, Requirements Working Group)
- Dr. Eldred Nelson (TRW)
- Mr. Bruce Peters (SDC)*
- Dr. Charles Rose (Case Western Reserve)
- Mr. Clark Weissman (SDC)

This report is an integration of the individual and collective contributions of the panel.

---

*Mr. Peters was with the Defense Intelligence Agency during the bulk of the study.

SECTION II

USAF COMPUTER SECURITY REQUIREMENTS

2.1 Introduction

This section reports the trends and problems in computer security that were identified by the panel's Requirements Working Group. The objective of the working group was not to develop firm coordinated command requirements for specific techniques or systems, but rather to identify the directions in which Air Force computer use is moving, and the bearing of these directions on computer security. For this reason, the Requirements Working Group was composed of working-level staff officers from Air Force commands that are major computer users. These officers presented descriptions of existing and planned computer usage and computer security problems within their commands. The Air Force commands that participated in the Requirements Working Group were:

Air Force Logistics Command	(AFLC)
Air Force Data Services Center	(AFDSC)
Satellite Control Facility	(SAMSO)
NORAD/Aerospace Defense Command	(NORAD)
Air Force Communications Service	(AFCS)
Air Force Global Weather Center	(AFGWC)
Strategic Air Command	(SAC)
Air Force Security Service	(AFSS)
Military Air Lift Command	(MAC)
Electronic Compatibility Analysis Center	(ECAC)

Sections 2.3 and 2.4 describe the trends and current problems, respectively, that were identified by the Requirements Working Group. Section 2.2 gives a brief summary of the range of system types and uses considered by the working group.

2.2 Range of Systems Considered

The systems planned or operated by members of the Requirements Working Group span a fairly broad range of functions. At one end of this range are systems that fully support general user programming in both batch and time-sharing modes. At the opposite end are relatively simple systems that perform only pre-specified functions, responding to user queries or switching messages. At an intermediate point in this range are systems that provide query or transaction processing to many online users and simultaneously support programming by a software maintenance staff.

Most computer systems discussed by the working group were of medium or large-scale size. Only these systems seem to have the capacity to make multi-user (and hence multi-classification) support practical. Current systems that present computer security problems operate on a mix of equipment supplied by almost every major manufacturer. Low and high levels of classification required by multilevel systems are determined by the nature of the using organization, but usually span a broad range of levels and special categories.

2.3 USAF Computer Usage Trends Affecting Computer Security

The following paragraphs describe the trends in Air Force computer usage that appear likely to have a significant impact on computer security problems of this decade. These trends are, in most cases, based on requirements and plans, rather than on existing systems. It is apparent that some systems cannot be built as planned, and some objectives will not be reached as long as present computer security problems remain unsolved.

2.3.1 Multi-Level Operation

Almost every member of the Requirements Working Group emphasized a need for multilevel secure operation of either planned or existing computer systems. The range of classification and clearance levels varies depending on the user organization and system application. In several cases, (AFDSC, ECAC, AFLC) requirements exist for operation at unclassified through secret levels. Other systems (SAC, AFGWC, MAC) are planned to operate at unclassified through top secret levels, while still others Tactical (TIPI) operate with all users cleared but with requirements to operate under strict need-to-know or special access controls. In all cases above where unclassified operation is mentioned, uncleared users and/or terminals with unencrypted communications are planned.

One significant trend was that the ratio of classified to unclassified data involved in a planned multilevel system can be quite small. Both MAC and AFLC estimated that "less than one percent" of system data is classified. However, because of the pervasive nature of security problems, both commands must go to considerable system-wide effort to attempt to provide effective security controls.

2.3.2 Open Operation

As was mentioned above, several planned systems are to provide processing of both classified and unclassified data with some users operating outside of a cleared environment. (That is, the users, their terminals or terminal communications are uncleared for any classified data.) Such systems are referred to as "open" systems and provide the would-be penetrator with ready entry points for his attempts to retrieve, alter, or destroy classified data. The growing user requirement for open systems is one of the most technically challenging trends identified by the Requirements Working Group.

2.3.3 Online Operation

Without exception, the systems described by the working group members will support online users at terminals. In some systems, the computer and its terminals will reside in the same building, while in others, terminals will be spread over a base, a metropolitan area, the country, or even the world. The planned systems will support large numbers of terminals (hundreds in the case of AFLC's Advanced Logistics System (ALS)) having varied security access privileges. The problem of providing security for this collection of terminals is compounded by the facts of online operation, which dictate small delays to user inputs and responses, and require that security checking overheads be reasonably low.

2.3.4 Transaction Systems

Many of the planned systems described to the Requirements Working Group are transaction processing systems — systems in which users may invoke only one of a known set of programs at a time. In such systems, users may take advantage of preprogrammed security weaknesses, but may not directly attack the computer and operating system with their own programs. Some Air Force systems planned or in development (ALS logistic processing, TIPI tactical information handling, AFGWC weather processing) are dedicated to transaction processing, while others (MAC's MACIMS reservation system, ADC Space Computation Center) provide both transaction processing and program development simultaneously. Users of transaction processing system feel that they should be able to use reduced security controls, since user threats have been reduced, but there is, at present, no general universal guidelines on adequate security controls for such systems.

2.3.5 Program Development

While some systems serve their users primarily in a transaction processing mode, almost every system, either planned or in being, is required to support some programming at some time. The bulk of the systems examined by the working group (all except the transaction-only systems mentioned above) require multiprogramming or program development with other system functions. In at least one system (AFDSC) there is a requirement for programming by uncleared remote users. In other cases, programming is restricted to a set (in ALS a very large set) of cleared development personnel. The presence of a program development workload on a processor handling classified data raises several/computer security problems: first, there must be some safeguards against a program (accidentally or deliberately) disabling security controls thus providing uncleared users with access to classified data; second, the transaction-only system typically exists in a changing mode and environment, and its operators must be constantly alert to assure that the security controls in the system are in fact complete and properly operating; finally, the program development must take place in a cleared environment (including use of secure terminals, where applicable) where the risk of external tampering with the system can be eliminated.

2.3.6 Networks

A final trend pointed out with considerable emphasis by the Requirements Working Group is the movement toward the establishment of large dispersed networks of related computer systems. AF Global Weather Center, for example, will interconnect several of its own computer systems. In addition, this interconnected complex will be tied to other weather processing centers and to the command control systems of (at least) SAC and MAC. SAC plans to tie several command control computers together, and may also interface intelligence processing systems. The MAC command control system, MACIMS, will be implemented as a network of WWMCCS computers. Plans are being formulated for a network to interconnect all of the WWMCCS computer installations. As networks of the types mentioned are developed, computer security problems, already difficult, become much more complex. For example, there is a possibility of one "untrustworthy" processor in a network collecting classified data from other processors by making apparently legitimate requests. Computer networks that have one or more nodes that can be accessed by users with clearances below the highest level of information in the network, constitute multilevel networks. The security threat posed by such operations is that, in general, the computer to computer communications are accepted as valid on the questionable basis that the other computer has a high security reliability. However, if control of a node can be exercised by a malicious users, the entire network may be compromised. In a network, it is essential that there be reliable security controls, that the nature of these be understood, and that the network does not inadvertently provide the means to bypass those controls. While there are growing requirements for interconnecting computer systems into networks the dimensions of the security problem are unknown. Much more information is needed on both the networks and their security requirements.

2.4 Current Problems

The previous section identified trends that lead to the computer security requirements of the future. This section outlines the major problems that arise today as a result of users' attempts to provide security with the products of current technology.

2.4.1 Off-the-Shelf Hardware and Software

Underlying most current users' problems is the fact that contemporary commercially available hardware and operating systems do not provide adequate support for computer security. While some limited protection is supplied in the form of memory protection controls, master and slave modes, and privileged instructions, experienced programmers have had little difficulty in penetrating off-the-shelf systems and retrieving desired data items. Certification attempts based on penetration have generally produced results leading to denial of certification. However, even an unsuccessful penetration attempt would not show grounds for certification, since the possibility of a yet undiscovered route into a large existing system is ever

present. Furthermore, so much of a current system is highly privileged (and potentially security-related) that there is a likelihood of a new security problem being introduced by the next update to the vendor's operating system.

Attempts to "patch" an off-the-shelf system for security tend to obscure penetration routes, but have little impact on underlying security problems. Existing systems have so many central privileged functions that the operating system becomes quite large and capable of concealing numerous flaws. Security packages may provide elaborate schemes for labeling output and handling user passwords, but do not effectively deter a programmer from accessing data as he wishes.

2.4.2 Ad Hoc Additions

Given the problems of current hardware and operating systems some users (AFGWC, AFLC) have been driven to the development of large software packages that mediate between applications programs and operating systems. Such packages are capable of providing a degree of security in a benign environment (no hostile programmers) but exact a very large price for storage space and execution time. These packages seem to offer little protection against a hostile programmer or possible underlying trapdoors and may be employed to protect (ALS) a small amount of classified data. Thus, their cost-effectiveness, at least, is subject to question.

2.4.3 Terminal Security

Given the operating system and hardware deficiencies described above, an organization (AFDSC) that wishes to support unclassified programming on a computer handling classified data has little choice but to do such processing in a secure environment. Creating this environment for remote terminals involves the use of cryptographic equipment that requires protection of its own. Thus, a computer user may find himself with a vault and cryptographic equipment for the protection of a terminal that processes only unclassified data. The cost of such a secure environment may be quite staggering — especially when multiplied by the number of terminals attached to a large time-shared computer system.

2.4.4 Media Declassification

Current technology does not provide for rapid and easy declassification of magnetic media (disks, drums) that have held classified information. Such media must be physically destroyed to guard against compromise of data that have once been stored on them. This destruction requirement represents a significant expense for CONUS computer users. In a tactical environment (TIPI) systems that may be overrun must have a safe, rapid method of declassifying media to avoid compromise of large quantities of data. Techniques for recording data in unclassified (encrypted) form or for rapidly clearing media would solve both the tactical and CONUS problems.

SECTION III

CONSIDERATIONS LEADING TO THE DEVELOPMENT PLAN

3.1 Background

Resource sharing systems are not currently widely used for multilevel classified processing because security and operations personnel are not convinced that they are secure against an internal user. This feeling is visceral — the technical issues generally being too complex to unravel in any particular situation. Nevertheless, their instincts are correct, and bolstered by the experience of programming errors resulting in protection bounds being accidentally breached. Frequently this prospect of accidental disclosure is cited as the reason for not performing multilevel classified processing on a system.

In fact, accidental disclosure on contemporary resource sharing systems occurs less frequently than in manual handling of classified documents, if the informal statements of security professionals are to be believed. Even so infrequent cases of accidental disclosure are generally not viewed as total disasters because of the generally valid assumptions that the person who is exposed accidentally to normally unauthorized information is benign.

If this is the case, why are not computers used for even simple multilevel classified operation? The simple answer to this is that the security bureauacy is concerned that even though all users are cleared for some level of information, the amount of investigation performed for the lower level clearances is significantly less than for the higher level clearances. Because of this, it may be possible for an agent to be placed in an organization and exploit in some mysterious way the concurrent processing of higher classified information with that to which he is authorized by his job and lower level clearance.

While some of the known expionage cases would indicate that there is at least as much to be concerned about from individuals already cleared but becoming untrustworthy (c.f. Martin and Mitchell), this receives attention primarily at higher clearance levels, especially those involving access to intelligence information.

When it is suggested that an agent-in-place threat exists, there is frequently a response indicating that the suggestion is a paronoid view, and not to be taken seriously. Yet it is precisely this threat that prevents multilevel secure computing on contemporary systems.

We have identified this threat as that of a malicious user. This term is more descriptive of the actual security concern, and avoids futile arguements over an individual's motives. We do not need to distinguish between a foreign agent or the misguided/disgruntled actions taken by an individual against the "establishment".

In addition to the experience of accidental disclosure, there has also been a number of successful penetrations of of systems where the security was 'added on' or claimed from fixing all known 'bugs' in the operating system. The success of the penetrations, for the most part, has resulted from the inability of the system to adequately isolate a malicious user, and from inadequate access control mechanisms built into the operating system.

In examining the broad threats to computer systems, it has been found useful to distinguish between external threats[1] and the internal (malicious user) threat described above. Both from the statements of the requirements working group and the panel's collective experience, it was found that the defenses against an external threat were by and large adequate and well understood. By and large the defense against external penetration is where the focus of computer security has been until now.

For many of the reasons discussed above, there is no adequate defense against a malicious user on most systems. It is the malicious user threat that provides the single largest barrier to providing multilevel 'secure' processing on most contemporary systems.

The requirements working group confirmed the panels assessment that this was the key problem to be faced, although as noted in Section II, they indicated that other important problems existed as well. While the requirements working group did not present coordinated requirements of their respective commands, their input was an informal expression of current and near-term problems being faced by working level staff officers and key civilians. It supported the experience and observations of the panel as a whole.

3.2 The Malicious User Threat

Having identified what is believed to be the key problem, the panel began to establish the requirements for a defense against a malicious user attack. In order to appreciate these points, it is necessary to understand some of the mechanisms used by a malicious user to achieve penetration of a system (an attack scenerio against a contemporary system is given in more detail in Appendix I). In contemporary systems, the attacker attempts to find design or implementation flaws that will give him supervisory control of the system. With supervisory control, he is then above to exercise parts of the operating system to access unauthorized classified data and return it to his own program in a way not anticipated by the operating system designers. Alternatively, he

---

[1]By external threats we mean those situations where it can be reasonably inferred that a computer system is the object of an expressed or implied intention on the part of unfriendly parties to acquire or modify information, or to deny its services to its legitimate users. The operative aspect of an external threat is that it is necessary to gain access to the system in order to carry out the threat. A malicious user (constituting an internal threat) already has access to the targeted system.

can either add to or temporarily replace parts of the operating system to give his program access and reference privileges not authorized to him. He may direct his attention specifically to the file containing the list of authorized users of a system (frequently containing the password(s) associated with each user). In any case, the attacker is able to reference any data or programs in the system.

As a malicious user is able to exercise more direct control over a computer through programming, he has the use of the computer as a tool to help his penetration and subsequent exploitation of the system. If he has a full programming capability using assembly or most of the higher order languages, he has the maximum possible user control of the system, and has available all but a few of the tools needed to aid him in his penetration. As the users capability is reduced by such means as forcing him to use interpretive systems, transaction processing systems and the like, his opportunities for direct control of the machine through his programming actions is correspondingly reduced because these tools are not sufficient for that purpose. His threat is reduced but unfortunately not eliminated through use of such techniques. Although the scope of actions directed to achieve penetration is reduced, he can still probe the system for exploitable design or implementation flaws using non-sequitor commands, false or 'nonesense' parameters, unanticipated interruptions, and the like. If the malicious user is a supported agent, he may merely exercise a 'trapdoor' placed in the system by another agent to gain access to classified data.

A number of the reasons that penetration attacks are possible are given below. A contemporary system provides a limited form of reference validation in the form of the memory protect scheme for the system. These schemes are designed to isolate the running programs from other programs and the operating system, and in general, work well enough on most systems. Because the schemes are so simple (either protection keys as those on the 360/370 or bounds registers in such machines as his 6000 series or the Univac 1100 series machines), they are generally applied to user programs only. The operating system, because it needs to reference all of the real memory on a system in exercising its control functions, most frequently runs with the memory protect suspended (i.e. in a supervisory or control state, where no checking of a reference is done) or with the memory protect set to enable the executive to refer to any memory without restriction (e.g. protection key zero in OS/360). While it would be desirable to confine references from the centralized service functions of an operating system to those parts of memory allocated to the user making the request, there is no convenient way on most machines to do so. Compounding this condition is the fact that many of the service functions made available to user programs are also used by the operating system in exercising its control of the system. In most systems it is not possible for a called service function to determine the identity of the caller and thereby 'interpret' the validity of the parameters or the service requested.

In conventional two-state machines, unrestricted addressing and privilege for executing I/O operations and setting memory bounds registers are associated with the supervisory state. Thus, the two-state machine is forced to enter supervisory state to provide the needed addressing capability, even to perform services not requiring privileged instructions, but requiring a capability to refer to data or instructions in

the callers workspace. Because of the all or nothing approach to memory protection, and because the simple bounds register technique forces programs and data to be bound together in contiguous locations, there is no convenient way to localize the referencing capability of an operating system service function.

The limited reference control provided by the memory protect schemes on most contemporary systems thus leads to monolithic, totally privileged executives with an unrestricted capability to reference any part of main or auxiliary storage. Because of the total privilege and unrestricted referencing capability of the executive, it is necessary for all parts of the executive to be designed and implemented correctly in order to assure that a system is proof against an attack by a malicious user. The sheer size of contemporary operating systems (on the order of 100,000 + instructions) and their complexity makes it virtually impossible to validate the static design and implementation of the system. When the dynamic behavior of the system is contemplated as well, there is no practical way to validate that all of the possible control paths of the operating system in execution produce correct, error-free results.

Because nearly all of the contemporary operating systems have so much of their code running in supervisory state, there are a large number of places a malicious user can attempt to attack a system. The primary points of attack include the I/O interface and the various system supplied service functions.

The attacks are possible because the operating system/hardware architectures tend to promote a monolithic totally privileged executive with unrestricted capability to reference any main or auxiliary memory locations. While it would be possible to design a more modular executive, the present design approaches (on contemporary hardware) provide the most efficient operation of the executive. A more structured operating system could be achieved on contemporary systems only by providing software controls (at considerable penalties in operating efficiency) to restrict references by the operating system. These conditions coupled with flaws or misconceptions in the design, and the fact that the operating systems <u>were not design to be secure</u>, provide a malicious user with any number of opportunities to subvert the operating system itself.

3.3 <u>Defense Against A Malicious User</u>

With the foregoing in mind, the requirements to defend against a malicious user can be better appreciated. These requirements are: A system <u>designed</u> to be secure, containing;

A) An adequate system access control mechanism

B) An authorization mechanism

C) Controlled execution of a users program or any program being executed on a user's behalf. We explicitly include the operating system service functions in this requirement.

It is the omission of design for secure operation and the lack of strict control of programs in execution that characterizes most contemporary systems, and which in combination with the size and complexity of the systems makes it impossible to conduct meaninful testing or certification to determine that the systems are secure. The key issues that emerge are program reference control and the correct design, implementation, localization and isolation of the security portions of an executive.

These requirements are amplified below. The adequate access limitation on users of the system and a defense control mechanism is needed to provide both a limitation on who uses the system, and as a defense against masquerading. By itself, it is not sufficient however necessary it may be.

The authorization mechanism is needed to represent to the system the user's clearances and need to know for data bases and programs. An authorization mechanism is an integral component of a system's security because of the role it plays in establishing what shared resources (data, programs, equipment) are permitted to a given user (or execution of that user's program or a program executed on his behalf). The requirement for controlled execution of a user's program (or a program being executed on his behalf) is merely a statement that requires the references made by the program to be those authorized for the user on whose behalf the program is being executed. In many situations, the user could have created the program as well, but this is immaterial. The combination of authorization mechanism (i.e. representation and attachment to a user program of the permitted referencing capability), and a system environment that controls the actual execution of the users program (or any program being executed on his behalf) to permit only the authorized references to be carried out is referred to as controlled sharing (of the systems resources). We made separate and explicit the requirement that the operating system being executed on behalf of a user is constrained to just the referencing capability of that user in order to firmly establish the point.

3.4 Security Models

In order to provide a base upon which a secure system can be designed and built, we recognize the need for a formal statement of what is meant by a secure system — that is a model or ideal design. The model must incorporate in an appropriate and formal way the intended use of a system, the kind of use environment it will exist in, a definition of authorization, the objects (system resources) that will be shared, the kind of sharing required, and the idea of controlled sharing described above. These elements should form a formal abstract specification of a secure system that can be proven to be complete, reflect real environments, and that will logically implement the controlled execution of programs.

Elements of a model of controlled sharing can be found in the work on capability models. These elements, and the concept of a reference monitor which enforces the authorized access relationships between users and other elements of a system form the basis for the recommended approach to the development of a secure resource sharing system.

In order to achieve the desired execution control of users programs, the concept of a Reference Monitor is used. The function of the reference monitor is to validate all references (to programs, data, peripherals, etc.) made by programs in execution against those authorized for the subject (user, etc.). The Reference Monitor not only is responsible to assure that the references are authorized to shared resource objects, but also to assure that the reference is the right kind (i.e., read, or read and write, etc.).

The notions of controlled sharing (the authorization mechanism and execution control) and the Reference Monitor are the central idea behind the recommended advanced development program. We have called the implementation of the reference monitor concept the Reference Validation Mechanism (RVM) — a combination of hardware and software that implements the reference monitor concept. In addition to these concepts, we add the additional principles that:

A) The reference validation mechanism must be tamper proof.

B) The reference validation mechanism must always be invoked.

C) The reference validation mechanism must be small enough to be subject to analysis and tests to assure that it is correct.

These principles specify the operating conditions of the reference validation mechanism.

The tamper proof condition is self evident. If the reference validation mechanism can be altered either programatically or manually, its integrity cannot be guaranteed, and no security certification of such a system could be derived.

The continuous invocation of the reference validation mechanism reflects that it must be applied to all programs including the operating system itself.

Finally the condition that it must be small enough to logically demonstrate that it is complete, faithful to the model, and correctly implemented is the same as saying that it must be capable of being proved to be correct.

3.5 Hardware Considerations

It is at the point of transforming these notions into a design that the efficiency of the reference validation mechanism becomes important. While a programmed interpretation may be acceptable for some applications, the requirement to support secure general programming suggests that hardware interpretation be used.

A computer hardware architecture based on descriptors provides the essential characteristics for implementing an efficient reference validation mechanism. The descriptor machines implement a virtual addressing capability. The association of a descriptor with each code or data object of a user's program including the current execution point of the program, provides an efficient mechanism for implementing a

reference monitor — that is a continuously invoked validation that all references made by a user's program or any (executive) program operating on a user's behalf, are authorized for that user. By incorporating the operating system service functions as implied portions of each user program (by providing descriptors pointing to these objects) and representing the authorized references for a given executing program as a table of descriptors, a precise control of the execution of both the user-supplied and implied programs can be achieved. Such an arrangement can eliminate the monolithic nature of the executive by restricting the reference capability of most of the executive to that authorized to a given user and represented by the descriptor table for his program. Under such conditions, the security sensitive portions of the executive can be reduced to the representation of authorization for a user (i.e. what programs and data a given user may have access to), programs to alter and maintain the representation, and the executive functions that create and manage descriptor tables. These functions are considerably less than an entire executive, and give rise to the expectation that only a small part of an executive will have to be demonstrated to have been designed and implemented properly in order to certify a system as secure.

It is not claimed that descriptor machines are intrinsically secure. Rather, they have the kind of architecture that provides efficient mechanisms that can be applied to the design of a Reference Monitor.

We noted above that descriptor machines implement virtual memory. Another approach to achieving secure operation is to implement a virtual machine (in the sense of CP-67). The main limitation of this approach is that the sharing of data and program resources may be too restricted for some applications. However, the architecture of such as system can provide adequate execution control of a user's program or any program or any program operating on behalf of that user, and may be a suitable base for building an executive for multilevel secure systems where only hardware sharing is required.

Descriptor machines that appear initially attractive as a basis for developing a system secure against attacks by malicious users include the Hewlett-Packard 3000 and Honeywell 6180 (Multics) systems. The Burroughs 6700, while a descriptor machine, leaves some descriptors accessible to user programs making it very difficult to assure that the descriptors have not been altered by a malicious programmed.

3.6 Obtaining A Secure System

It is clear that the reference validation mechanism described above is not a model of secure computing. It is a device to provide containment of programs in execution, and as such, is at the heart of any implementation of these ideas. Surrounding this particular element are others that collectively make up the security part of a system. These include the authorization mechanism, the access control mechanism, and for government applications, methods to record and properly label files and printed material with the proper security markings.

The first step in applying those ideas is to create a model or ideal description of a secure system that incorporates the various elements described above. It integrates the intended use, use environment, the threat, a definition of the desired authorization, and the notion of controlled sharing and reference monitoring to produce the specification of an "ideal" secure system. Different kinds of use, use environments, and threat will produce different models of what constitutes a secure system. We have suggested the following:

Use: Multilevel, General Programming

Use Environment: Open Use

Threat: Malicious User

Authorization: Unclassified — Top Secret

The thrust of the modeling is to make sure that all of the necessary elements have been considered, and are properly reflected in a statement of the specification of a secure system. The model must be stated in terms that permit logical determination, that the desired security objectives will be achieved if indeed a system based on the model is produced.

After the modeling has taken place, it is then possible to begin to develop the design for the security portion of a system, which we call the Kernel. The security Kernel design incorporates the reference validation mechanism, access control (to the system) and authorization mechanisms. Further, it will probably incorporate the administrative programs to represent and maintain user and program authorizations, since it is anticipated that the authorizations will change frequently. At this stage, the hardware architecture becomes important not only in achieving efficient implementation of the RVM but also in how other parts of the Kernel will be handled.

During this stage, we would expect the application of both formal and informal techniques to continuously evaluate how well the Kernel design meets (conforms to) the ideal specified in the model. Based on the design, it is expected that it will be necessary to go through several levels of implementation, again employing the best certification techniques available to be sure that the result conforms to the model.

The last stage is the integration of the Kernel with other existing software on the prototype system. In order to minimize the costs of the prototype development and to take advantage of the existing software, the panel recommended conducting the development on the HIS 6180 (Multics) System.

We are not unmindful of other real technical problems that arise in connection with processing multilevel classified information. However, many of these are procedural in nature. Solutions to these problems without solving the malicious user problem merely provides the illusion of security and simultaneously a real danger of significant compromise.

3.7 The Engineering Development Plan

The recommendation for the Engineering Development Plan comes from the observation that the cost of providing security for crypto equipment for terminals used in classified processing can exceed by many times the cost of the equipment itself, and that the effective accessibility of resource sharing systems becomes limited if the cost of using the system is increased due to the need to physically protect crypto equipment. In addition to reducing the direct costs of classified processing, a low cost secure computer terminal is necessary for unclassified processing in those situations where the risk of degraded service due to external penetration is evaluated as high. In general, the availability of such a terminal will make it possible to perform classified processing is <u>any</u> suitably cleared area with no additional costs incurred due to special protection of crypto equipment.

The crypto multiplexer is similarly motivated as a means of making it possible to secure (with separate KG's) the terminal end of a link without incurring the cost of terminating each link into a separate KG.

The file encryption techniques development is directed to solving the problem of handling classified removable media (tapes and discs), and the problems of attempting to provide rapid destruction of classified computer based information.

The computer security handbook is an attempt to provide in one place, a collection of useful techniques that can be used in developing and operating a secure system. It is envisioned as containing standardized security practices associated with operating both single level and multilevel secure systems.

3.8 The Related Advanced Development Plan

This plan, described in section VI, is presented as the only alternative the panel sees to operating all current systems as single level closed systems until the results of the Advanced Development program become assimilated technology. This plan recommends the early application of the results of the modeling activity conducted under the Advanced Development Plan to the development of secure Query/DMS based transaction systems, and to provide the basis of evaluating the feasibility of selective reimplementation of operating systems of USAF inventory machines.

In all candor, the latter application is expected to show that it is <u>not</u> economically or technically feasible to reimplement such systems. Nevertheless, the problem is of such urgency, that we believe no avenue should be arbitrarily shut off without having been examined.

With respect to the payoff of using the security modeling to guide the development of multilevel secure transaction systems, we are more optimistic. As a consequence, we have recommended the development of a multilevel secure Query/DMS system to serve as the nucleus of a variety of transaction systems. Although such systems would

remain vulnerable to 'trapdoors' placed in the operating system, we believe revised modes of operating would effectively eliminate this vulnerability.

3.9 Exploratory Development Plan

The Exploratory Development Plan contains a variety of semi-independent topics that support the development of multilevel secure computers for the Air Force. These include studies of alternate hardware configurations for secure computing and techniques development for administrative and procedural aspects of security.

These disproportionate size of the recommended program is a reflection of the fact that there has been no on-going program of exploratory development in the past. The panel believes that a vigorous program of exploratory development is necessary in order to bring a continuing stream of techniques and approaches to apply to the problem.

SECTION IV

THE DEVELOPMENT PLAN

4.1 Introduction

The development plan is based on applying the concept of a reference monitor and the accompanying operating principles to derive a model of a secure computing environment. This model will be used to develop efficient designs for hardware and software mechanisms needed to provide a centralized protection mechanism, for a variety of applications, including the ultimate objective of a secure open-use multilevel system supporting general programming.

4.1.1 Security Principles

The technical threat in contemporary systems posed by a malicious user is that because the systems are produced using ad hoc security rules, a penetrator will find a design or implementation flaw, or induce a 'trap door' situation to obtain supervisory control of the system. An analysis of various successful programming attacks reveals that their success is dependent on the penetrators program or the supervisory system acting on his behalf making a reference to program or data not authorized for that user. While a kind of reference validation (in the form of memory protect features) applies to user programs in many contemporary systems, this validation is often not applied to the supervisor. Complicating the problem is the lack of viable hardware mechanisms in most contemporary systems to apply the reference constraints of a user's program to the supervisory system operating on his behalf.

It is hypothesized that a system secure against internal malicious threat from a programmer can result from employing a reference monitor to validate all references to programs or data according to the access authority of the user on whose behalf the program is executing. In concept, the reference monitor mediates each reference made by each program in execution by checking the proposed access against a list of accesses authorized for that user. The reference monitor concept is implemented as a reference validation mechanism. Accompanying this concept are the operating principles that:

a. the reference validation mechanism must be tamper proof.

b. the reference validation mechanism must always be involved.

c. the reference validation mechanism must be small enough to be tested (exhaustively if necessary).

These principles, vigorously applied, can result in integrating all of the system security controls for a system into one hopefully small portion of the operating system code. If this portion is than implemented correctly (i.e., without any programming flaws), and cannot be altered by any other part/function of the system the security

concern of how the rest of the system or any user program is implemented is focused on the access authorization(s) permitted to the user(s) or programs.

The concept of a reference monitor and the operating principles described above are derived from the work of Lampson[1], Graham and Denning[2] and others in developing capability models, and represent the most viable approach to developing certifiable secure systems. An abstract model of secure computing, incorporating these principles is needed to identify the security sensitive parts of operating systems, and to provide a basis for evaluating the adequacy of a given operating system design. By having a model of secure computing, it is possible to develop integrated designs of protection mechanisms that incorporate in one place in a system all of the technical security mechanisms needed to provide a computing environment secure against the malicious user threat. The realization of a design from the model involves determining representations of access privileges, and a set of primitive operations needed to maintain the representations for the users.

(Because the model is oriented to solving the problem of the malicious user, it does not deal with the problems of physical security, system access authorization, communications security etc. These important facets of system security must be dealt with using existing technology).

Because the reference monitor concept implies interpretation of each reference made to determine the validity of the attempted access, efficient mechanisms for this interpretation are required if the concept is to be viable.

4.2 <u>Outline of the Plan</u>

The development plan to achieve secure, open use systems has as its objective the development of a prototype of a secure computing system derived from a model of an 'ideal' secure computing system. The model development is considered to be the first step of the development, as it establishes the technical requirements of such a system. Based on the model, the design of a 'security kernel' incorporating the access control, reference validation and security related functions is to be undertaken and validated. Parallel with and contributing to the model development and kernel design are systems studies evaluating the applicability of various systems organizations to the problem. The result of all the studies culminate in the prototype development which will implement the security kernel on a suitable system for a specific USAF 'customer'.

The funding estimates shown below, for this and subsequent sections are the collective judgement of the panel as to the amounts needed to obtain various parts

---

[1] Lampson, B. W., "Dynamic Protection Structures," Proceedings 1969 FJCC

[2] Graham, G.S. and Denning, P.J. "Protection-Principles and Practices," Proceedings 1972 SJCC.

of the program. Like any estimate, these recommendations can be challanged on specific points. In general, they reflect the experience of people who understand the subject matter, the issues involved, the probable degree of difficulty of the task. All of the panel members had the experience of working on or managing similar tasks. It was assumed that the people doing the work described in this report were familiar with the issues involved, as well as the technology of operating systems, computers architecture and the interaction of these technologies in the design of resource sharing systems.

4.3 Development of Model

The objective of this task is to provide a complete model framework for open-use, multilevel secure resource sharing system(s), supporting general programming. The model will be based on the concept of validating each and every reference, and the application of the operating principles of continuous invocation at all times, self-protection, and logical completeness.

This development is an extension of the capability models of Lampson and others to incorporate elements found in Government classified information processing, and generalizes the notions of access to include people, terminals, and other non-central aspects of a computing facility. Successful completion of this task will provide a complete description of the essential aspects of security in computer systems, and be applicable in a number of subsequent tasks. The task(s), schedule and funding are shown below.

(All Funds Shown in $ Millions)	FY					
Task	73	74	75	76	77	78
Develop Model of Secure Resource Sharing	.15	.15				

4.4 Security Kernel Design

As part of the secure systems development program a key ingredient is to convert the secure systems models into operating systems components in order to determine a number of aspects that at present are only surmised. These aspects include the amount of code that is crucial to the security protection provided by the operating system and the degree of complexity this code represents.

The objective for a security kernel design is to integrate in one part of an operating system all security related functions. This is for the purpose of being able to protect all parts of the security mechanism, and to apply certification techniques to the design. The kernel design transforms the abstract security model into computer hardware and software elements that represent the model.

Earlier we indicated our preference for a certain kind of hardware for accomplishing these studies. We used the term 'descriptor-driven' virtual machine to describe that

hardware, although it must be emphasized that not all machines with descriptors are suitable nor are all machines suitable that can claim (to some degree) virtual processing capability.

The key ingredient of the type of machines that are important to the achievement of a secure operating environment (user programmed open-use systems) is the use of descriptors to represent 'name spaces'. That is, the descriptors constrain the addressing of a user to only those parts of memory representing information referred to in his program and impose a hardware mediation on all references for instructions or data. The hardware mediation of all references on using descriptor mechanisms is essentially that required for security protection mechanisms. Further, descriptors provide an efficient mechanism for checking the type of reference permitted for each user (e.g., Read, Write, Execute).

The task, schedule, and funding for the Security Kernel design is indicated below:

	FY					
Task	73	74	75	76	77	78
Develop Security Kernel Design	.1	.15	.1			

4.5 Systems Studies

The panel strongly recommends the implementation of at least one fully documented and provable (certifiable) secure operating system, both to serve as an instance of what is required to achieve this level of security in a real system and to assist in the translation of the abstract modeling into a set of specifications that can be used to procure future systems of a similar kind. Although the panel favors the use of descriptor-driven systems for providing general use, open-use secure systems, it is by no means the only avenue that can be explored, and it is recommended that the development program include exploratory and developmental studies in the areas of functional distribution of operating system functions over physically segregated machines and the "Shared Machine" approach of CP-67, VM/370.

The former scheme, in some ways, modeled on the CDC 6600 family of equipment, is an alternate approach to providing secure operation since it physically isolates the user from security sensitive parts of the operating system itself. It is of interest to note that this approach to the work is that currently underway at University of California at Berkeley on the PRIME system.

The Berkeley work and the security kernel approach are both dependent on the same thing: being able to isolate security sensitive portions of the operating system. In the latter case, the expectation that descriptor interpretation hardware can be used to maintain proper separation of the security sensitive portions of the operating system is reasonable, since if the separation does not work at this level it will not work at any of the lower levels and the system is invalid. Similarly, the notion of functional segregation of portions of the operating system, particularly the security sensitive parts,

depends on being able to identify these parts and provide them a degree of physical protection by the functional and physical separation employed in that approach. We believe the development program should evaluate components of this work primarily because it appears that the notion of distributed function systems is reasonably probable for future systems development in the next five to ten years.

The "Shared Machine" notion of CP-67 and VM/370 provides a control program that shares the hardware of a physical computer among its users in a secure fashion that lets each user have a different "virtual machine" operating an operating system of his choice. It is a potential solution to the problem of sharing physical resources among users with disjoint information requirements. The data and file sharing capabilities of this approach are somewhat primative; operating at the level of shared virtual media rather than shared logical files.

Based on an evaluation of the various systems, the decision as to which approach to use in development of a prototype can be taken.

The objective of the systems studies is to investigate alternative system approaches as a means of obtaining open-use multilevel secure systems. These studies will test the generality of the model of secure resource sharing by providing implementation alternatives and evaluate the applicability of other current systems work to USAF problems. Because of the latter point, the studies are planned over a three year period, with the major effort occurring in the first year.

The tasks, schedule, and funding are shown below:

	FY					
Tasks	73	74	75	76	77	78
1. Investigate Partitions of Operating System Functions to Isolate Security Related Functions (SRF) Including those Identified by Model	.1	.05	.05			
2. Define Interface Between SRF and Other System Function	.05					
3. Evaluate Other Systems Approaches as Implementation Alternative	.05	.05				
Totals	.2	.1	.05			

4.6 Prototype Development

Although it is possible that the results of the systems studies will favor other approaches for prototype development, it is anticipated that the prototype development

will take place on a single integrated system. The objective for the prototype development is to provide a multilevel secure resource-sharing system as a demonstration vehicle, to determine what must be included in systems design and procurement specifications for subsequent system purchases, and to tailor the development for a specific USAF customer. It is anticipated that the development will be done on an existing descriptor-driver virtual machine such as the ARPA-Sponsored Multics System in order to make as much use of previous work as possible and because of its demonstrated usefulness. Further, the prototype will serve as a base for developments in security surveillance, and implementation certification and recertification techniques. The tasks, schedule and funding are shown below:

		FY					
	Tasks	73	74	75	76	77	78
1.	Select and acquire hardware base for prototype	.05					
2.	Identify Security Related Functions (SRF) present in software base that will be replaced by Security Kernel design		.1				
3.	Implement Security Kernel on target machine		.5	.35			
4.	Reorganize Balance of System Software to Interface with Security Kernel		.2	.2			
5.	Certify Security Kernel Implementation		.2	.1			
6.	Test and Evaluate			.3	.3	.2	.1
7.	Develop Procurement Specifications for Secure System			.2	.2		
8.	ADP Support	.2	1.0	1.0	1.0	.5	.3
	Totals	.25	2.0	2.15	1.5	.7	.4

SECTION V

SUPPORTING ENGINEERING DEVELOPMENT

5.1 Introduction

Two areas of security developments have been identified as contributing to cost reduction in the design of information systems. These are a handbook of computer security techniques and the development of secure peripherals for use in resource sharing systems.

5.2 Handbook of Computer Security Techniques

The purpose of this element is to make available to the USAF in readily available form the most current security technology which can be employed in the design and acquisition of systems. The product of this activity will be an unclassified, continuously maintained handbook which can be used by System Program Offices and contractors alike to achieve a uniformly high standard in the acquisition of system security. Ultimately, the handbook should contain the following data (or direct references in lieu thereof):

a. Criteria for classifying systems with respect to their security requirements.

b. Threats to security and types of security failure.

c. Design practices to defend against specific types of failure and threats.

d. Known techniques and devices for access control, user identification, file labeling, system audit and surveillance, etc.

e. Recommended security practices in the design, management, installation, and operation of secure systems.

f. Recommended practices for the acquisition of secure systems, including model specification clauses and certification methods. (The state-of-the-art does not permit meaningful incorporation of this section at the start, but it is a development objective.)

The initial effort should be directed to organizing, compiling and issuing a version of the handbook which will contain all currently available information on the topics listed above, and which will be in a form that permits its maintenance into the indefinite future. Following this is a continuing effort responsible for maintaining the handbook to reflect the current state-of-the-art.

The tasks, schedule and funding are:

	Tasks	FY 73	74	75	76	77	78
1.	Initial Handbook-Compilation	.15					
2.	Handbook-Maintenance		.1	.1	.1	.1	.1
	Totals	.15	.1	.1	.1	.1	.1

5.3 Secure Peripherals

The inclusion of the development of secure peripherals as part of this plan is a reflection of the present high costs of physically securing terminal sites to protect them and their associated crypto equipment from tampering. These costs, and the costs associated with interfacing a large number of secure lines to a central computer severly limit the application of resource sharing systems to current USAF problems. The prospect for spontaneous developments in these two areas is almost non-existent. Coupled with these developments is a program to eliminate the major existing problems of physically protecting magnetic media containing classified information.

5.3.1 Secure Computer Terminal For Office Environments

The objectives for a secure office environment (computer) terminal is to design and develop a capability to provide inexpensive encryption between a remote terminal and a computer system which

a. is virtually transparent to the terminal operator (re: operation, physical protection, keying, etc.), and

b. adds minimally to terminal cost; i.e., makes maximum use of terminal's resources such as clock, power, enclosure, etc.

The advantages of an inexpensive/transparent/integrated terminal encryption device are:

a. More extensive use of remote terminal capability of secure ADP systems.

b. Protection against inadvertent spillage of classified information (if all users can be cleared and terminal areas can be physically protected to the level of information handled by the ADP system).

c. Depending on its application, it could provide terminal/user identification/authentication.

The plan includes the development of six (6) prototypes in order to provide enough copies for test and evaluation.

The Tasks, schedule and funding for this development are:

	Tasks	FY 73	74	75	76	77	78
1.	Define terminal hardware operation, transmission characteristics, communication model, and physical environment of the terminal	.1	.05				
2.	Develop appropriate encryption device, and method of use. Design and implement terminal hardware incorporating device (6 prototypes)		1.4	.8			
3.	Test and Evaluation of Terminal			.1	.15		
4.	Develop Procurement Specifications				.05		
	Totals	.1	1.45	.9	.20		

5.3.2 "Multiplexed" Crypto Concentrator

The objectives of this development is to design a capability to provide encryption for all links between a computer and its remote terminals, and other computers (with one, or some very small number of crypto "devices") in order to

a. minimize cost, operator controls, space and other environmental requirements,

b. provide more than one secure communications path via the same transmission link, primarily on a time multiplexed basis.

An ADP system generally has a large number of communications lines terminating at the computer. It is highly desirable that devices needed to secure any or all of these links do not considerably increase the cost nor impact the operations of the CPU facility. Additionally, transmission speeds are very slow relative to the speed of the computer and I/O, and a maximum utilization of these resources indicates a form of the store and forward approach as one possibility which would permit sharing of the encryption device. This device may also be designed to provide user/CPU/terminal authentication.

The Tasks, schedule and funding are shown below:

	Tasks	FY 73	74	75	76	77	78
1.	Define the communications model, response delay, transmission speeds, variables transmission techniques, number of terminals to be served, their speeds, etc.	.1					
2.	Determine applicable design concepts, extent of interface with and control by CPU. Develop prototype design.	.1	.1				
3.	Develop prototype model and interface it with a CPU.		.1	.3	.1		
4.	Test and Evaluation				.2	.1	
5.	Develop Procurement Specifications				.1		
	Totals	.2	.2	.3	.4	.1	

5.3.3 File Encryption Techniques Development

The objective of this development is to design a capability to encrypt any and all information stored on magnetic media in order to be able to handle the media as unclassified. This capability should not noticeably affect computer thruput or processing times; it should make maximum use of existing features of the computer; and it should be virtually transparent to, and independent of, the system user.

A secure file encryption mechanism would alleviate a major existing problem of physically protecting magnetic media containing classified information. Depending on the technique developed, it might also protect against inadvertent spillage and file access errors. The principal benefit of this work appears to be in use in tactical systems or systems that exist in hazardous environments and which are exposed to capture.

The Tasks, schedule and funding are shown below:

	Tasks	FY 73	74	75	76	77	78
1.	Analyze file encryption requirements to determine implementation technique (special hardware, software (in CPU), file processing computer)	.05					
2.	Define crypto technique interfaces to CPU and device controllers	.145	.2				
3.	Implement file encryption capability		.3	.2			
4.	Integrate into ADP System and Test			.15	.15		
5.	Develop Procurement Specifications				.05		
	Totals	.15	.50	.35	.2		

SECTION VI

ALTERNATE ADVANCED DEVELOPMENT PLAN FOR INTERIM SOLUTIONS TO CURRENT PROBLEMS

6.1 Introduction

The main thrust of the recommendations of this panel is to focus on obtaining solutions to obtain certifiably secure open-use multilevel application of resource sharing based on designs dervied from an abstract model of secure computing. However, current systems security problems are so compelling that it is necessary to consider interim solutions to these as well. Our reasoning is based on the observation that no matter when a multilevel system secure against the malicious user threat is achieved, there will be a continued requirement to provide multilevel secure operations on systems already in the USAF inventory. Thus, until the design of multilevel secure systems becomes part of assimilated technology, and are commonly available from a variety of manufacturers, there will be a continuing security problem with existing systems. At present, the only safe step is to operate such systems as closed single security level systems. However, in a number of cases, this is an expensive proposition, since it implies that large numbers of people will have to be cleared even though their jobs require no access to classified material merely because they will be using a system that contains some classified information. Two systems, currently under development, that fall into this category include the Logistics Command's ALS, and the Military Airlift Commands MACIMS. Another major system that may have similar properties is the WWMCCS. Although unclassified processing by uncleared users is not a WWMCCS requirement; the ability to support both the National Command Authority and applications 'local' to the WWMCCS site indicate the possibility of varigated clearances/classifications for the different kinds of work.

In examining the current USAF environment, there are essentially three courses of action that can be taken:

a) Operate USAF systems as closed, single level systems

b) Develop restricted-capability multilevel systems, based on the models of secure systems developed as part of the Advanced Development Plan

c) Reimplement the operating systems of selected contemporary computers based on the security model(s) developed under the advanced development plan.

Of course operating systems as closed single level systems are required in a number of USAF systems. For these environments, there is no need to do any more than is currently being done. However, for the many other real and potential applications environments that do not require the single security level operating environment, the single level approach is costly and unresponsive. Largely because the systems require some form of multilevel operation, the panel has considered and recommends development in two necessary areas. A useful restricted capability system is one that

supports multilevel on-line query and data management operations. Here the prospects for success are quite good, and the development coincides with a number of current and planned USAF system applications. The recommendation for development of a secure higher-order language (HOL) only system described below is motivated by the consideration that the operating environment is very similar to that needed for a secure Query/DMS system, and the design of a reference validation mechanism suitable for one would encompass many of the requirements for the other. The other consideration behind this recommendation is the fact that a general user programming capability is often needed in most environments, and this method appears to offer the best interim solution to providing that capability.

The degree of threat posed by a malicious user in this kind of environment is a function of the amount of programming he can do. For example, if the malicious user can only (legitimately) use an on-line transaction-oriented Query and DMS, his capability to affect the operation of the system is limited by the intrinsic capability of the tools he can use. Most transaction-oriented systems do not provide the malicious user with sufficient tools to take over control of the system; he cannot attack the system with his own programs. He may be able to gain unauthorized access to classified data by exploiting a pre-programmed weakness due to careless design or implementation, or planted as a 'trap door' in the application or in the programming and operating systems supporting the application. The security threat posed by this mode of use depends on whether the application is designed in such a way as to assure that each user is fully controlled in all actions he may take on the system. In addition both the application and the programming and operating system for the hardware supporting the application must be implemented by trustworthy (cleared) personnel in order to preclude the possible inclusion of 'trap doors'.

Because transaction-oriented systems are so prevalent in USAF applications, we recommend that the model be used as the base for developing a secure multilevel data management and query system as an interim way to obtain secure multilevel transaction systems. It appears feasible to augment the existing hardware and software controls in contemporary systems with a programmed reference validation interpreter, subject to the risk that trap doors have been inserted in the application or the software for the base machine. It may also be possible to use the same technique to support the general use of one or more of the higher order programming languages (only).

While any realistic assessment of the trap door threat would have to conclude that to date there is no evidence of malicious placement of trap doors in contemporary system software, there is no technical problem to doing so. Under present modes of operation where installations accept operating system updates and even whole revisions of an operating system without question, there is little doubt that the targeted system(s), could be induced to accept and install a trap door modification to their operating system.

Further, as long as present day commercial computer hardware is used to base even transaction-oriented systems, the complexity and size of the operating system programs running in supervisory (control) state leaves the practicability of analyzing them (or their revision) for trap doors in doubt.

The recommendation regarding reimplementation of current systems was the most controversial in the panel. Although the panel was not optimistic about its possible success, it was believed to be an important enough issue to warrent at least the feasibility investigation. The panel was agreed that reimplementation or repair should only be undertaken after a model specifying what security was being provided was completed, and the reimplementation designed to implement the model.

6.2 Data Management/Query and Higher Order Language (Only) Systems

Although contemporary systems are unable to support general programming in a secure multilevel mode, the development of an abstract security model is directly applicable to the development of secure multilevel Query systems, and other kinds of transaction systems. It is also possible that it can be applied to general programming only in restricted languages (e.g., FORTRAN or JOVIAL) although the efficiency in such application may be unacceptable in some environments. Described below are two areas of a secure open-use system. These are important because they can be realized on systems currently in USAF inventory, and will provide secure multilevel information processing operations suitable for many current USAF applications.

6.2.1 Security Requirements of Query Systems

The security of a Query system is dependent on the:

- access control and reference validation mechanism,
- limitations on expressive power of the query language,
- interpretive execution of the query language.

Access control in a query system must restrict each user to the portions of the data base to which has has been authorized, while the reference validation mechanism restricts the user to the portion of the data base he is authorized.

A query language should be limited in its expressive power in three ways:

- It should be able to reference data elements by name only and not by storage location or by relative identification.
- The functions on the data elements that can be invoked should be limited to those needed for the application; they should be referenced by name and it should not be possible to algorithmically construct complex functions.
- It should not provide direct links to a host programming language.

Interpretive execution ensures that the user cannot directly modify the machine code, thus complying with the tamper-resistant and continuous invocation principle. This is in contrast to the compiler type programming systems, in which a user writes a program in a higher order language, after which program is compiled and returned to the user as a machine language program. The user may then have the opportunity to modify the machine language program before it is executed. With interpretive execution, the user cannot do that.

It is believed that the abstract security model will form the basis for designing an interpretive reference monitor that will be a powerful reference validation mechanism for this kind of application.

6.2.2 <u>Security Requirements for HOL-only Systems</u>

A higher order language (HOL)-only system is one in which the user of the system can program only in one or more approved languages that are translated into machine code by an approved compiler, and which are executed in an approved environment (called the run-time package) that controls the reference capabilities of the program. Within these constraints the user is permitted to write any program he can express in the language(s) given him.

The major vulnerability to be guarded against in HOL-only systems is the possibility that the user (programmer) of the system may escape from the higher order language to enter or execute arbitrary machine code <u>of his choice</u>, and defeat or bypass the run-time package.

In the discussion to follow, we refer to FORTRAN because it is a common language and serves the purposes of illustration. The primary technical problem is whether the FORTRAN user can break out of the FORTRAN envelope into data areas and thus be able to execute arbitrary instructions planted in the program as data.

In order to break out of the FORTRAN envelope it is necessary to be able to execute references outside of those defined by the FORTRAN program itself. These would include references beyond the upper or lower bounds of an array branching to an unlabeled area, or being able to overlay code with data. In general, the ability to write beyond the defined area for code or data in FORTRAN is sufficient to break out of the FORTRAN confines into the domain of the real machine.

Considering these problems, we can establish the following requirements for a secure higher order language (only) system.

a. There is a rigorous separation of code from data (of all kinds, including constants).

b. All references to data (of all kinds) are validated to assure that no code locations are accidentally or otherwise obtained.

c. All transfers of control are validated to assure that the control point sought lies within the code area only, and only to recognized labels.

d. All input-output transfers are validated to assure that data read or written is that authorized to the user, and does not overflow the boundary of the array or vector being referenced.

An interpretive reference monitor based on the abstract security model may be able to validate such accesses and provide a secure HOL only multi-level programming capability on contemporary equipment.

The tasks, schedule and funding for these developments are given below:

	FY					
Tasks	73	74	75	76	77	78
1. Develop Reference Validation Interpreter Design	.1	.2				
2. Apply to (Implement) DMS/Query System for contemporary machine	.2	.3	.2			
3. Apply to Higher Order Language (only) Programming environment on contemporary machine	.1	.2	.1			
Totals	.4	.7	.3			

6.3 Repair of Current Systems

The basis for recommending this particular development is the fact that present systems are not technically secure for applications where programming is one mode of use of a system. It is not just the so-called open-secure systems that are of concern. Rather it is the fact that nearly all of the systems that support general programming have an inadequate technological base to provide even minimum need-to-know on security controls.

It is also evident that even if a fully certified secured system or systems were presently available that replacement of the existing inventory of computer systems and the applications contained thereon is not feasible, nor do we believe that it is feasible at any time in the foreseeable future under similar circumstances. As a consequence, it is necessary to at least examine the steps necessary to provide security on existing systems even though these steps may be properly viewed as a stopgap measure.

The secure computing model can provide a basis for examining the design and implementation of contemporary computing systems and assessing the degree of effort required for their repair. The objective of this effort is to survey key contemporary systems to determine whether it is economically feasible to redesign and/or reimplement their operating systems to provide secure computing environments to the applications based on these systems.

The panel cannot overemphasize its belief that "patching" of known faults in the design or implementation of existing systems without any better technical foundation than is presently available, is futile for achieving multilevel security. We wish to distinguish, however, between the patching problem and the possibility of selective re-implementation of portions of an operating system to eliminate known security deficiencies and to provide a better technical foundation for the development of more secure systems for some environments. We do not see any method to provide the level of security desired by the Air Force for many of its systems through any simple technique or simple fix. It is also evident that re-implementation of nearly all contemporary systems would be necessary in order to provide even the minimum level of privacy necessary to implement need-to-know controls in all applications involving classified information. It is recommended that only those systems in widespread use be considered. Obviously, a prime candidate for such a system would be the WWMCCS using the Honeywell 6000 series equipment.

6.3.1 Reimplementation

The most obvious solution for an existing system with security problems is to rewrite the software following the principles of the abstract security model. This is a problem of unknown magnitude and requires a feasibility study to determine if it should even be undertaken.

In this connection it may be possible to reduce the cost of reimplementing current operating systems to overcome their security deficiencies by applying the techniques and technology of virtual machines. Specifically, it would be necessary to examine current operating systems to determine whether or not the availability of an applique of hardware (such as the Bolt, Beranek & Newman paging box and monitor for the TENEX System based on the PDP-10) or the possibility of modification of the hardware to create a segmented virtual memory system would have any effect on either the extent and/or the cost of reimplementing those portions of the systems judged to be deficient in security. Techniques to be examined also include the possibility of creating independent virtual machines for each process (program/user) on a system, and confining reimplementation modifications to those necessary for applying access controls on such things as the file system and the portion of the operating systems devoted to managing the real memory of the computer.

Unfortunately, the panel does not hold much hope for the ultimate success of this development. However, the problems with contemporary systems are so limiting, that as a minimum the feasibility investigation should be undertaken. Tasks associated with this development are:

a. An analysis to determine the scope of reimplementation of a selected system based on the results of the abstract security model. This analysis would be directed to identify those components of the operating system that would require reimplementation in order to eliminate generic flaws of implied sharing, incomplete parameter checking, too rapid a response to user initiated interrupts and the like.

b. If the analysis indicates reimplementation is feasible, determine whether various techniques such as virtual machines, distributed systems or descriptor driven virtual memory systems will provide aid to reimplementation.

   Even if the analysis does not indicate the feasibility of selective reimplementation to achieve secure operation, the effort is not wasted. The results are applicable to new designs, need-to-know controls, and as a basis for evaluating proposals for upgrading equipment.

c. Estimate the cost and effects of reimplementation of the selected operating systems both with and without the hardware aids chosen for those systems.

d. If feasibility is still indicated, undertake the development of the necessary hardware and/or reimplementation of those portions of the operating system deemed necessary to provide the technical security level desired.

The tasks, funding, and schedule for this alternative plan for <u>one</u> system are shown below.

(All Funds Shown in $ Millions)	FY					
	73	74	75	76	77	78
1. Analyze key current system for extent of redesign or reimplementation	.1	.2	.2			
2. Conduct hardware modification/applique studies		.2	.2			
Subtotal for feasibility investigation (1 system)	.1	.4	.4			

Assuming that the analyses indicate feasibility of repairing or reimplementing a system, the additional tasks are:

3. Design and Install Hardware Modifications or Appliques			.3	.5	.2
4. Redesign and Reimplement Key System			.4	.8	.4
Subtotals for Reimplementation	—	—	.7	1.3	.6
Totals (1 system)	.1	.4	1.1	1.3	.6

Detailed projections for subsequent systems are not shown, but are estimated to be 80-90% of the effort shown above for each additional system.

6.4 Security Surveillance

An area of security techniques development that is in danger of being overlooked is that associated with the role of procedural controls in establishing and maintaining secure operations.

A major objective is to achieve a security surveillance capability on secure systems. The emphasis on a security surveillance capability is a reflection of the desire to detect breaches of security or penetration attempts. Unfortunately, the audit schemes developed around existing facilities (mostly accounting oriented) in contemporary systems are too inflexible to provide either surveillance or a damage assessment capability to systems security personnel.

A security surveillance capability is related to the instrumentation of a system. To date the emphasis on (hard or soft) instrumentation has been for system performance measurement. While it can be seen that a security surveillance capability requires many of the same points of measurement, the security surveillance differs in what is recorded, and more importantly how it relates the measurement to the real world of users, terminals, communications lines, etc. Further, from a security surveillance viewpoint, while <u>all</u> possible measurements are not of interest all of the time, all possible measurements will be of interest (not all at once) at some time. Secure systems must be capable of supporting a variety of security surveillance audits at different levels of detail simultaneously. For example, it must be possible to monitor (record) each direct and induced transaction on behalf of one or more specific users, while maintaining a running record of the use of several of the communications links, while recording all transactions (by each user) against the files on a particular physical storage device, and to be able to vary the mix and focus easily on a day-to-day or shorter time basis.

To be determined are the most promising way of relating a user, terminal, physical device, etc. to the measurement points, and how to vary as a function of the level of surveillance being maintained, what is recorded upon reaching a given (program) measuring point. While it seems reasonably clear that both hardware and programs can be provided measuring points at little cost, the best way (or even alternate ways) to achieve the desired security audit capability is not yet well understood. Because of this, the funding for security surveillance is included in the corresponding section of the Exploratory Development plan.

SECTION VII

COMPUTER SECURITY EXPLORATORY DEVELOPMENT PLAN

7.1 Introduction

The computer security research plan complements the development plan by outlining studies leading to alternative systems designs, new techniques, improvement of operations, and in general, a better understanding of the problem(s) of secure computing. To the maximum extent possible, this part of the plan has focused on research directly applicable to computer security problems, and has avoided recommendations in security-related areas such as fault tolerant computers, advanced programming languages, and the like.

The research topics are less structured than the preceding development plan, however. The contributions of each topic to the various problem areas is shown in the table, Figure 7-1. Each topic is presented independently.

7.2 Systems Architecture Research

The objective of this research is to examine alternate configurations of computers, and new computer organizations that may lead to secure operations. There are three topics specifically recommended in this area: Distributed Systems, 'Software First' Systems, and Internal Encryption.

Distributed Systems Studies have already been included as part of the development plan, but should continue to be examined because they represent an alternate future direction for computers, particularly as logic costs continue to become an insignificant part of a total system's cost. One model of particular interest is the possibility of implementing a file control system in a minicomputer separate from the CPU.

In virtually all present day computing systems, one of the weak points is the file handling capability. Such a file handling system could either directly manage the media; that is, the various tapes and disks that would be connected to the minicomputer, or else it could implement these activities together with all of the other operations required to form a modern file system. In other words, the machine represents a separated piece of programming logic. The file system looks hardwired to the main machine although it is internally programmed.

Another approach is to design a minicomputer-based system that could simulate certain parts of an operating system. The most obvious choice would be a minicomputer which would interface with virtually any large scale computer. It would take the place of the operator's console and would relieve the operator of many of the decisions that he now must make in a large scale modern computing system. This is important in a security environment in that it is now possible on many systems to fool the

Research Item / Contributions to	Development of Secure Open-Use Systems	Cost Reduction in System Design	New Technology	Procurement and Production	Security Theory	Secure Operations	Certification and Recertification
Secure Networks	x		x		x		
Security Models			x		x		
Security Software Engineering				x			x
Security Surveillance	x		x			x	
Certification Techniques	x		x				x
Architecture Research	x	x	x		x	x	
Data Integrity and Reliability			x			x	x
Automatic Classification			x		x	x	
Magnetic Media			x			x	
Computer Aided Integrated Design	x			x			x

Figure 7-1. Relationship Between Research Items and Security Problem Areas

operator into taking certain actions that will violate security restrictions. At the same time such an operator's system could also form the basis of a security audit system which collects the major statistics as a mere by-product of all the information the operator's console receives. Such a system could be more computer independent than many other potential solutions to this problem. The programming of such a system could be quite independent of the particular main system that it is serving. It can be made specific to the explicit system by means of descriptor tables which are used to

define the various types of operator interactions that can take place on the particular system to which it is connected.

Research on direct execution of higher order languages ('software-first' systems) is already being supported by the USAF as a means of reducing programming costs, and increasing the efficiency of storage use, especially in air and spaceborne systems. The work recommended here is an investigation of the kinds of security problems that would be solved by using software-first machines. On the surface, it would appear to be an alternative method of isolating users from the balance of the system, and further could have the added benefits of reduced programming costs. Should the initial investigations confirm this view, a prototype can be developed using one of the available microprogrammed systems.

The possibility of internal encryption of computer programs and data was first advanced in 1966 prior to the Defense Science Board Task Force on Computer Security. Since that time it has received sporadic attention. It appears that it is possible to apply this technique either as an applique or as an integral part of the design of computer systems.

Originally it had been anticipated that the use of the technique could act as a cosmetic coverup for the many known deficiencies of operating systems of that time. However, it turns out that the technique is not as broadly applicable as first thought because the major effect it has is to render the effects of reaching outside of one's memory space into other parts of a shared resource memory system unproductive due to the fact that the information thus recovered cannot be read because it is encrypted. A better understanding of what the computer security problem is and particularly the focus of attacks on systems indicates that this particular problem is basically solved with any of the current memory project mechanisms.

Virtually none of the attack methods exploit any direct attempt to gain access to information outside of a user's address space. Use of internal encryption would have some beneficial effect on the scavenging problem and reduce the overhead of having to overwrite or otherwise control the use of mass storage used for work files. It is not clear that this benefit would be outweighed by the increased overhead penalty involved in operating a suitably high grade crypto algorithm for providing such protection.

The area of principal benefit appears to be in application to tactical systems or those systems that exist in environments which are exposed to potential overrun by hostile forces. In view of the increased complexity of the operating system to attempt to cope with these situations by such means as automatic purge routines and the potential inability to exercise any significant portion of a purge in any extreme situation, we believe internal encryption is worth examining.

Other topics for investigation include the functions and design of a free standing minicomputer based Security Officers console.

The role of microprogramming in establishing and maintaining a secure operating environment requires careful analysis. While it is possible that the technique could isolate (from a malicious user) security related code it may be illusory if the microprogram can be manipulated.

Finally, we recommend a continuing program of security related systems studies in order to provide a continuous evaluation of new techniques and technology.

The Tasks, schedule and funding for these investigations are shown below:

	FY					
Tasks	73	74	75	76	77	78
1. Distributed Systems						
a. General Studies	.1	.1	.1	.05	.05	.05
b. File Computer	.05	.1				
c. Automated Operators Console	.05	.1				
2. Internal Encryption						
a. Security Impact Studies	.05	.05				
b. Design Studies		.05	.10			
3. General Studies	.15	.15	.25	.30	.30	.20
Totals	.4	.55	.45	.35	.35	.25

7.3 Networks

The networks of interest can be characterized as two or more digital computer systems interconnected through digital communication lines. Many current operational computer systems satisfy this model. The subset of specific security interest are those computer systems which interchange information without any need for human intervention -- systems which originate queries automatically (triggered by user requests, sensor or environmental event, alarm clock, schedule) and which receive responses from other nodes in the network. Usually when sensitive information is being exchanged, the nodes of the network rely on secure communication lines. Though considerable financial resources and management attention are drawn to the communications security aspects of networking (an important but well understood technology) the security problem of computer networking is not a communications problem but another more sophisticated instance of multilevel computer operating system security.

Currently, most secure computer systems achieve their security integrity by prohibition of multilevel and multi-compartment security operation. The computer is operated at a single, appropriately high security level for its needs, with all personnel and operating procedures controlled within the USAF/DoD established security

framework. Networking ties two or more of these computer systems together; more often than not, systems dissimilar in equipment, configuration, purpose, management, and security control procedures. An example of the networking problem is the connection of the SAC SATIN network with AUTODIN network for both the receipt and transmission of information. Conceptually, the network can be viewed as a "supra computer system". The network security requirements then are different than most of its members because the "supracomputer" operates essentially as a multilevel, multi-compartment, multi-user computer system. The network's security vulnerability is that each network node (i. e., the computer system operated by a participating agency) is unprepared for multilevel, multi-compartment use by users over which it exerts limited, if any, control. Furthermore, the problem often goes unrecognized since management erroneously assumes security integrity because the supracomputer interconnections are via secured (often crypto) communications lines.

"Third-party" identity checking is a problem aggravated by computer networking. In its simplest form, user A (first party) authenticates his identity to his computer system B (second party) by password or other techniques.

Because of the nature of the A-B dialogue, computer system B makes a network request to computer system C (third party). C now has the problem of determining the authenticity of B's identity or worse, of A's, and further that B and A are authorized for the requested information. Solutions to this problem must be general and satisfy "nth-party" authentication. Network solutions patterned after user A - computer B authentication schemes, common in most current multi-user systems, collapse because such schema would require all nodes to have rapid access to an unmanageably large and frequently changing network user-profile data base. This is an unsolved problem in general, deserving of research attention. Specific operationally accepted (but weak) solutions have been realized by restricting authentication to just two parties, e. g., A-B, B-C; and trusting to network procedures that all users of a given node are scrutinized by that node with a network-acceptable authentication method.

In essence, networks provide a unique security problem totally unrelated to the communications media forming the network. At present, too little is known about the security problems of networks or even how all USAF networks are interconnected. In conducting the requirements investigation, we found a number of intrasystems connections that not only formed networks, but that interconnected two unrelated networks, often at different classification levels. To better understand, and control security in networks, the following tasks, schedule and funding are recommended:

	Tasks	FY 73	74	75	76	77	78
1.	Detail and document existing (and planned) USAF networks configurations	.1	.1				
2.	Conduct Security Analysis of existing and planned networks to identify security control elements		.1	.05			
3.	Define subsystem (network) and network security requirements and mechanisms for appropriate security control		.15	.15			
4.	Perform network data aggregation studies	.05	.05				
5.	Devise third party authentication techniques	.1	.1				
6.	Network Security Plans and studies		.1	.25	.20	.20	.20
	Totals	.25	.6	.45	.20	.20	.20

7.4 Abstract Security Models

Although the development plan is predicated on being able to adapt the work in capability models into an abstract security model that can be used as the basis for design of access control mechanisms that will properly bound all users, there is no assertion that the model perceived for the current development is the only one possible, nor even one that embraces all aspects of a secure computing environment. It is clear, however, that an abstract security model is an absolute requirement if certification of systems is to ever occur. Without adequate models of security of computer systems, it is not possible to design secure systems. For that reason, we recommend a strong continuing program of research in abstract security models. Topics that should be undertaken include development of models that represent resistance to inadvertent disclosure of classified information (possibly by identifying program and hardware elements that require redundancy); models that deal with the classification of merged or extracted and regrouped information, leading to techniques of automatic classification of data; methods of security rating of information systems according to the level of information they contain and the degree of resistance to attack they represent.

In addition to development of new or alternative security models, the research program should refine and evaluate the models by applying them to operational systems (current or planned) to demonstrate the cost/benefit advantages systems based on the models over current approaches. The schedule and funding are shown below:

	FY					
Task	73	74	75	76	77	78
1. Modeling	.25	.25	.25	.25	.25	.25
Totals	.25	.25	.25	.25	.25	.25

7.5 Certification Techniques

Assertion that a Security System represents a solution to the security problems of resource-sharing computer systems will require certification that the system's hardware, software, and procedures -- both in the design and implementation provides an acceptable (to the certifying agency) secure mode of operation. The techniques that can be used today to certify a program are quite primitive; however, interest and concern for program certification research is increasing very rapidly. Even independent of the computer security problem, it is reasonable to expect that extensive funds for research in this area will be made available by the government agencies funding other advanced development. Since progress in techniques for certifying programs will also be vital to a computer security effort, the agencies funding advanced development should be encouraged to fund several major projects aimed at the general problem of program certification techniques. Serious work to bring the more formal certification techniques to the point of being practical for large programs will require very large scale integrated development efforts.

The certification required by the development plan involves proving that the security kernel is always invoked, is tamper-resistant, and validates each and every reference in the system. In essence it must be possible to demonstrate that it is complete, performs correctly, and does not perform any function not specified. Note that because of the centralization of the security functions, it is not necessary to certify the entire operating system.

One formal approach[1,2,3] to the development of reliable software is called the "Proof of Correctness." Essentially it involves writing formal specifications for what one wants to guarantee about a program. Then the specifications and the program code are translated into a statement of formal logic such that if that mathematical statement can be proven, then any execution of the program will satisfy the specifications. The proof has to be based on a large number of axioms or assumptions about the operating environment of the program and the programming language semantics. Thus at best this technique could only guarantee reliability relative to these assumptions. Furthermore it may not be easy to write clear specifications for security in a formal language. In any case this technique will need extensive development before it can be useful for computer security since thus far no proofs have been given for programs longer than a few hundred lines of code. This is at least one to two orders of

magnitude smaller than what would be needed even for a minimal security system. However, this approach has the potential of ultimately providing a much higher level of confidence in a system than could be achieved by the usual testing techniques.

As a way of validating a design, a variation of this approach, called "structured programming", has been developed by Dijkstra[4] and applied to developing an operating system. It is a top down approach in which a program structure is built one level down at a time. At each level, the next lower level of the structure is denoted by a name (or abstraction) assigned to it. For each level, a proof, in which the denotation of each name denoting a lower level is considered to be correct, is constructed. The resultant program has a well defined structure and most of the errors are removed in the process of proving correctness at each level.

Since a secure operating system must have a correct implementation, the operating system for a secure system should be developed using the structured programming approach; the security features should be given specific proofs; and the system should be thoroughly tested, until the persons responsible for certifying it are satisfied that it has an acceptable secure mode of operation.

Because the techniques that can be used today to certify those aspects of a secure computer system that are specific to computers are quite primitive, there is a need to develop more precise and complete techniques. Also since a secure computer system must not only have its design certified, but, after each update and modification, the system must be recertified, the techniques must be useable also in installation certification and recertification. Some Techniques which have shown promise are:

Automated Verification Aids. The thoroughness of the verification of operating systems has been limited in practice by the amount of testing that can be performed within the limits of the time and money available and by the lack of any model to determine when testing is complete. The approach recommended for this development, that of defining a model of secure computing, and locating all security related functions in a single simple and small portion of the system will minimize the certification problem by localizing what has to be certified to a hopefully small portion of a system.

In order to achieve certification of a design, it will be necessary to develop further those tools already found to be useful in program testing. Automated verification aids have recently been developed for application programs which automate several of the tasks involved in test planning, test production, and test execution and analysis. This accelerates the testing cycle and reduces the amount of labor required, and aids in increasing testing thoroughness. Some of the techniques used which are appropriate for extension to the verification of operating systems are:

- automatic analysis of the anatomy of an operating system - i. e., identifying all "testable segments" (sequence of code that has only one input and one exit) and all transfers between segments,

- quantifying the thoroughness of the testing by instrumenting the operating system to measure the fraction of segments and transfers exercised in each test and cumulatively over a series of tests,
- identifying the portions (segments and transfers) not tested in a series of test cases and indicating the input data needed to exercise them,
- identifying all entrances to sensitive areas of an operating system,
- identifying all interrupts and the logical paths they can initiate,
- investigating other characteristics of operating systems for suitability for automatic analysis and quantitative measurement - e.g., time dependent processes.

Although the techniques described in the preceding paragraph can significantly increase the thoroughness of testing of application programs and operating systems, they do not address an important aspect of data management systems - the correctness of design and implementation of the data structure. Flaws in the data structure may open the door to penetration of highly sensitive parts of the data base. Accordingly, techniques of automatic analysis of the anatomy of data structures and quantification of the thoroughness of testing of data structures should also be developed.

Because the certification process does not stop with the design, but must also assure the correct implementation of the kernel design, the certification techniques development spans several years. The tasks, schedule, and funding recommended are shown below:

		FY					
	Tasks	73	74	75	76	77	78
1.	Investigate and Define Design Certification Techniques	.1	.1	.1			
2.	Apply Design Certification to Security Kernel		.1	.05			
3.	Develop Automated Program and Data Structure Testing Techniques	.05	.1	.1	.05		
4.	Apply Automated Testing Techniques to Prototype System		.05	.2	.05		
	Totals	.15	.35	.45	.1		

7.6 Security Surveillance

General Observations

Security surveillance is defined here as the use of servomechanisms for maintaining continuous control over the security state of a system. Its functional purposes are:

1. To detect security-related events (i. e., system behavior which constitutes or precipitates security incidents or violations).
2. To collect, record, reduce and analyze data regarding event detections in order to invoke an appropriate compensatory procedure (e. g., exception processor, alarm or correction mechanism).
3. To generate reports for security personnel review and damage assessment.

Technical Development

In discussing the technical development of security surveillance capabilities and techniques, attention is focused upon the functions of instrumentation, measurement, compensatory procedures, reporting and integrity.

Instrumentation. There is a two-fold problem associated with instrumenting a system for security surveillance. What shall we detect? How shall we detect it (them)? These questions are complicated by the fact that while all possible system events are not interesting all of the time, all possible system events are interesting (but not simultaneously) some of the time.

Measurement. Once data regarding security related events is generated, it must be collected, recorded, reduced and analyzed to determine their security implications. Techniques are needed by which the mechanisms of measurement are dependent upon the security significance of the data to be analyzed.

Compensatory Procedures. A much neglected aspect of security surveillance has been the range of actions or procedures to be invoked upon detection of a security incident. Techniques are needed to refine the corrective role played by the monitor personnel by use of explicit cues, instructions (if necessary) and checks.

Reporting. Tied closely to considerations of measurement is the question of how to inform those individuals responsible for the security of the systems about the results of security surveillance. Reporting techniques are needed which consolidate information regarding security incidents and compensatory actions into a form that is meaningful and that minimizes the effort required for review and follow-on actions. In addition, techniques are needed which allow the security manager to retrieve information (perhaps including snapshop dumps) required at a given instant.

Integrity. Here the critical issue is how the survellience capability is protected and what measures are present (or assumed) to prevent the capability from being bypassed.

Assuming that the problems of instrumentation, measurement, compensation and reporting can be resolved, there remains the difficult matter of combining these elements into a viable whole for a particular system. This process of integration and operation requires careful consideration by the system developer of fitting security surveillance into its proper niche in the ADP system as a whole.

Research and development is required to build an extensible prototype of a comprehensive security surveillance capability for purposes of testing technical concepts and techniques. Based upon this work the prototype could be adapted to existing systems and/or designed into future systems.

The Tasks, schedule and funding are shown below:

	FY					
Tasks	73	74	75	76	77	78
1. Instrumentation and Measurement Studies	.2	.1				
2. Compensatory Procedures Studies		.15				
3. Reporting Techniques		.15	.1			
4. Surveillance Systems Design			.1	.1		
Totals	.27	.4	.2	.1		

7.7 Computer-aided Integrated Computer System Design Environment

The cottage-crafted approach to computer systems design and development is being gradually eroded by the demands of integrated circuit production. No longer is it possible to make simple design adjustments while fabricating a prototype as was the case when discrete components were used. Because of the costs involved, there is a requirement to get the design correct before committing it to hardware. To date, there has been no equivalent economic pressure to improve the design of software. Over the past ten years, there has emerged approaches to systems design that integrates the programming and computer performance requirements into integrated designs. However, these approaches have been disjoint combinations of manual and automated techniques, with greater emphasis on the manual aspects. The objective of this program is to provide a computer-aided design environment for the hardware and software of a secure computer system that will increase the probability of certifying a given design over hand design and implementation techniques. It will provide syntactic and analytic checks on a developing design as well as a formal framework for applying the techniques of structured programming and proof of correctness (of programs or hardware). Such a system (Project LOGOS at Case Western Reserve University) is in development under DOD funding. It should be able to impact the design

of production systems in the 1977-78 time period. The major (single) task is transfer of technology through the design, implementation, certification (or the extent possible), and documentation of a secure computer system responsive to a USAF requirement using the LOGOS design environment. To insure transfer, the work should be performed by USAF personnel under the guidance of LOGOS personnel.

	FY					
Task	73	74	75	76	77	78
1. Perform single integrated hardware/software development using LOGOS		.1	.3	.5	.2	.05

7.8 Miscellaneous Research Topics

This section contains a collection of topics that defy classification. Included are topics of Data Integrity and Reliability, Automatic Classification, and Magnetic Recording Media Research.

7.8.1 Data Integrity and Reliability Study

The objective is to devise a methodology of incorporating redundant and/or error checking/correcting into a data structure such that the host computer can determine if the data structure is in a consistant configuration. This methodology should also aid in recovering a data structure after it has been damaged by computer system malfunction. The methodology to be devised is not concerned with the contents of the data structure but whether or not the data structure itself is in allowable configuration. The objective is to flag and/or correct an inconsistency in the data structure which may go otherwise unnoticed until that portion is accessed or modified.

7.8.2 Classification Aids

The objective of Classification Aids is to develop automatic methods and techniques for assisting users in the classification of their data transactions. Recent advances in English text processing systems make it feasible to consider application of such technology to automatic classification. These efforts at Massachusetts Institute of Technology, University of Wisconsin, Stanford Research Institute, Bolt, Bernak and Newman, and System Development Corporation (SDC) also show promise of inferential data base construction. Lastly, set-theoretic approaches at SDC have shown practical application to high water mark upgrading. These techniques will address the problems of:

a. Lexical analysis of text.

b. Automatic downgrade of classification based on:

   1. Data subset abstracting

   2. Elapsed time

c. Automatic upgrade of classification based on:

   1. Data Set Aggregation
   2. Data Set Implication by Inferential Techniques
   3. High Water Mark

7.8.3 Recording Media

The objective of this topic is to research the technology to identify recording media which would satisfy ADP peripheral storage requirements and yet not possess the undesirable property of magnetic remnants; and/or discover and develop techniques for controlled, automatic and rapid degaussing of magnetic media when unpredictable changes in (removal of) physical security occur. If a successful file encryption technique is developed and implemented, these efforts would have major application for core (primary) memory; alternatively, it might serve in place of the apparent need for file encryption.

The funding and schedule for these research tasks is:

		FY					
	Tasks	73	74	75	76	77	78
1.	Data Integrity	.1	.15	.20			
2.	Classification Aids	.2	.2	.25	.20	.20	.10
3.	Recording Media Studies	.05	.05	.1	.1	.1	.1
	Totals	.35	.4	.55	.30	.30	.20

SECTION VIII

COST SUMMARY

8.1 Advanced and Engineering Development Plans

The Advanced and Engineering development plans are shown together because they represent the main thrust of the panels' recommendations. The output of both of these programs includes prototype hardware.

Cost Summary For Recommended Computer Security Program(s)
(All Amounts Shown in $ Millions)

	Fiscal Year 73	74	75	76	77	78	
I. Development of Secure Open-Use System Prototype							
1. Develop Model of Secure Resource Sharing	.15	.15					.30
2. Develop Security Kernel Design	.1	.15	.1				.35
3. Systems Studies	.2	.1	.05				.35
4. Prototype Development (includes ADP Support)	.25	2.0	2.15	1.5	.7	.4	7.0
TOTALS	.70	2.4	2.3	1.5	.7	.4	8.00
II. Supporting Engineering Developments							
1. Handbook of Computer Security Techniques	.15	.1	.1	.1	.1	.1	.65
2. Secure Office Environment Terminal	.1	1.45	.9	.2			2.65
3. Multiplexed Crypto Concentrator	.2	.2	.3	.4	.1		1.20
4. File Encryption Techniques	.15	.5	.35	.2			1.20
TOTALS	.60	2.25	1.65	.90	.20	.10	5.70

Although the primary emphasis of the panels' activities were directed to developing the program to obtain a multilevel secure system, described in the advanced development plan, the current problems were so evident that it was felt necessary to address these as well.

The size of the cost estimate for the exploratory development program is due to the inclusion of all of the important items that could be perceived by the panel. In effect, it reflects the fact that security technology is complex and pervasive, and that there has been too little effort in this area in the past. Many of the items should have been accomplished long ago, (at significantly lower costs) but have not due to the low level of interest shown in the past.

8.2 Related Advanced Development and Exploratory Development Programs

Because they are outside the main development stream, a related advanced development to provide interim solutions to current problems, and an exploratory development program in computer security are shown separately. Since the interim solutions development is addressing current problems, the funding for these items should come from existing programs. The figures shown are our estimate of what the effort will cost. The exploratory development program is directed to provide a continued influx of techniques and technology bearing on the problem of secure computing systems.

Cost Summary for Related Developments and Exploratory Development Program
(All Amounts Shown in $ Millions)

	Fiscal Year						
III. Developments for Interim Solutions to Current Problems	73	74	75	76	77	78	
1. Secure DMS/Query Systems	.4	.7	.3				1.4
2. Repair One Current System	.1	.4	1.1	1.3	.6		3.5
TOTALS	.5	1.1	1.4	1.3	.6		4.9

	Fiscal Year						
IV. Exploratory Development Plan	73	74	75	76	77	78	
1. Hardware Architectural Studies	.45	.70	.70	.75	.45	.20	3.25
2. Systems Technology	1.15	1.95	1.95	1.05	.85	.75	7.70
TOTALS	1.60	2.65	2.65	1.80	1.30	.95	10.95

REFERENCES

1. R. W. Floyd, "Assigning Meanings to Programs", Proceedings of Symposia in Applied Mathematics, Vol. XIX, Mathematical Aspects of Computer Science, American Mathematical Society, Providence, Rhode Island, 1967, 19-32.

2. R. L. London, "Computer Programs Can Be Proved Correct", Theoretical Approaches to Non-Numerical Problem Solving, Proceedings of the Fourth Systems Symposium at Case Western Reserve University, R. B. Banerji and M. D. Mesarovic, (eds), Springer-Verlag, 1970, 281-303.

3. B. H. Liskov and E. Towster, "The Proof of Correctness Approach to Reliable Systems", Mitre Technical Report 2073, 9 March 1971.

4. E. W. Dijkstra, "Notes on Structured Programming", Technische Hogeschool, Eindhoven, August 1969.

APPENDIX I

SECURITY THREATS AND PENETRATION TECHNIQUES

BACKGROUND

The traditional statement of security threat has had the classical objectives of:

a. information recovery

b. manipulation of information

c. denial or degradation of service.

While any of these threat objectives may be the ultimate goal sought by a penetrator they do not really describe the basis for concern about computer security. In this paper we will attempt to outline the nature of the security threat against computer systems, give some instances of the types of attacks used, and a scenario of an attack in order to give the reader more familiarity with how the problem appears to a penetrator and why the simple security measures devised in benign environments do not accomplish the desired results.

We use the concept of security perimeter to define the limits of control over the process of producing a system. This concept is important since it is usually in connection with the misunderstanding of the importance of having as broad a security perimeter as possible that the disagreements regarding the degree of security offered by a particular system arise. As an example, it may be possible to implement the technical controls in order to control reference to program and data objects in a system. These controls may be fully understood and certified, yet if the system in which they exist is produced by unreliable people, there is no assurance that the underlying hardware and software upon which the controls may have been built are themselves in any way secure. While to some this may seem to be an extreme view, what it indicates is that the security perimeter extends only as far as the security controls themselves. As a consequence, there is an understandable reluctance to certify systems where control over the production process is itself unknown. It is because of the unknown and potentially malicious aspects of the production process that the security problem of contemporary systems is as complex as it is.

SOURCE OF SECURITY THREAT

The objective of the development program is to provide a secure computing system where the procedural, physical and clearance controls over the user population are not necessary or even possible. It is given therefore that a hostile third party has direct programming access to a targeted computing system. It is the direct programming access to a computer system that constitutes the principal security threat. In the sections that follow, we will attempt to illustrate how this threat can be exercised by enumerating classes of attacks and where appropriate, the generic flaws in the design or construction of an operating system that are exploited.

CLASSES OF ATTACKS

Implied Sharing

It is a property of many contemporary operating systems that the monitor portion of the operating system will share memory space with user programs, either as work space or as a convenient place to put information associated with that user program. This condition often arises from a deliberate design policy invoked to charge the individual users directly for resources that they use. If the user requires file operations or other kinds of system resources, it is appropriate to maintain the information and the work space for the operating system working on behalf of that user in an area that will be uniquely chargeable to that user. Because the workspace is shared, but in a mode not normally available to the user, the implementors of the operating system often are careless with regard to the state in which their workspace is left after receiving a user request.

In one contemporary operating system, the monitor uses such a workspace to read in the catalog of authorized users of the system along with their passwords as part of a search for data requested by a given user. This function is necessary in order for the system to determine that the requests are properly formed and authorized to the user making the request. Upon finding the condition that a request is improper, the monitor returns control to the user program making the request, with an indication of the nature of the error in the request. However, it does nothing about the information remaining in the shared workspace. As a consequence, the user can now refer to the workspace and obtain from it other user identifiers and authenticators (passwords) which he can then use to masquerade to the system.

The same operating system has provision to record the state of a running program at convenient restart points as "checkpoint" dumps. The checkpoints are recorded on a file specified to the system by the user where it is then available to that user for manipulation. The user can then cause the program to be restarted using modified state information that accesses different data than that originally specified. In an earlier version of the particular monitor in question this could result in the user gaining supervisory state control of the system. (This particular condition has since been repaired).

While there are a variety of countermeasures to this class of attack, it is interesting to note that the situation upon which the attack depends may well have occurred deliberately due to design decisions on the part of the operating system designers. Further, it is important to identify all instances of implied sharing in order to apply the appropriate countermeasures.

Scavenging

The scavenging problem can also be called the 'unerased blackboard' problem as far as contemporary computer systems go. What this attack exploits is the fact that

work files and workspace in general (cf. Implied Sharing) is not erased after use, even after the program using the space is completed.

Taking advantage of this is a simple matter and admits a variety of attacks. In its simplest attack, a program is written that specifies large tables (as workspace). If the operating system does not clear the workspace assigned, the program then can read what had previously been written there, print it, and search for useful information.

Most operating systems designers recognize this particular problem especially in connection with the higher level languages such as FORTRAN because it could create an implied set of initial conditions on the programs written in those languages. As a consequence uncleared main memory is not as commonly found as in some of the other memory media.

Another type of scavenge uses the same approach on files instead of main memory. For this scavenge, one writes a program that defines the requirement for large file space. The system will generally allocate the space at the time the program is readied for execution. One then opens the work files for reading instead of writing, and reads the (previous) data.

In general, it is impossible for the operating systems to determine the intent of the programmer in any particular sequence of actions he may take.

Results from scavenging are not predictable as one can well imagine, however, on some systems, this kind of scavenging has resulted in retrieving user identifiers and passwords (from batch run control cards) as well as complete programs, data files and the like. If nothing else is available to a penetrator, scavenging is an acceptable source of substantial amounts of information.

As in the implied sharing vulnerability, there is a very simple countermeasure to counteract the effects of scavenging. The reason this has not been implemented in contemporary systems is that the overhead associated with erasing all file and work space after its use is high, and most users are unwilling to pay this penalty. Regardless, about the only currently effective countermeasure is to erase the workspace after it is used. The fact that most systems have no provision for doing this indicates that scavenging is still a useful attack method for most systems.

Incomplete Parameter Checking

The major weaknesses of contemporary operating systems occurs at the interface between the system and the user. This interface is present in order for the user to exercise the various centralized functions and services provided by the operating system to all users (e.g., I/O operations, program initiation, date and time, etc.). Users call operating system functions in a manner similar to subroutine calls, providing the details associated with a call as a parameter list. The bulk of the parameters are (expected to be) pointers (addresses) to information within the callers assigned space. While much attention is given in an implementation to validating the

operating system call parameters, the multiplicity of implementers almost guarantees that one or more important checks will be overlooked.

By supplying addresses outside of the space allocated to the users program, it is often possible to get the monitor to obtain unauthorized data for that user, or at the very least, generate a set of conditions in the monitor that causes a system crash.

In one contemporary operating system, one of the functions provided is to move limited amounts of information between system and user space. The code performing this function does not check the source and destination addresses properly, permitting portions of the monitor to be overlaid by the user. This can be used to inject code into the monitor that will permit the user to seize control of the machine.

A further example occurred in another contemporary system. The monitor expected parameters to come from the users space and did in fact check that this was so. (The only check it made was whether the parameter appeared to be in user space). If the parameter came from system space the parameters were accepted without further question. This situation occurred because the call(s) involved could come from users or other parts of the operating system (e. g. , a call to allocate more space on a temporary basis). The monitor had no way of distinguishing which case it was handling except through this simple check of the source of the call. From this it was possible to deceive the monitor into eventually returning control in supervisor state to a user program.

The attack was developed along the following lines. First an instruction transferring control to a predetermined point in the users program was loaded into a register. Next, a system call was made that caused the register(s) to be stored (saved by the system) in system space. Upon return of control, another system call was made that used as a transfer point an <u>implicit</u> parameter (an address) stored in the user space that had to point to a location in system space. If a user space address was supplied, the parameter check would catch it and abort the call (and the program). Naturally, the address supplied was the location in the register save area where the transfer back to the user program had been planted by the previous system call. All parameter checks passed, and control was returned to the user in supervisory state giving him control of the system.

The incomplete parameter checking attacks are less easily countered than the previous examples because the existence of the vulnerability relies on what must be considered design or implementation flaws. These flaws do not mean that the functions being exercised do not operate correctly. Rather it means that their interactions with other functions are so uncontrolled as to produce unknown (to the implementor) side effects. Contemporary operating systems provide manifold opportunities for this sort of attack if only because their sheer size precludes design, certification, and development by a single (or a few) knowledgeable individuals.

Asynchronous Interrupt Attacks

This class of attack is directed to exploiting how a system handles asynchronous interrupts, and attempts to bypass one or more security related controls by injecting an unanticipated interrupt in the middle of an execution of that control. As an illustration of this kind of situation, many contemporary systems provide a user up to three chances at logging on correctly before summarily rejecting his attempts. The limitation imposes (what is believed will be intolerable) delays on a user attempting to exhaustively enumerate all possible user-id's and log-on authenticators (passwords). Since such exhaustive enumeration is ultimately controlled by the communications line speed, automatic sign-off acts to limit the number of attempted log-ons possible in a given period even if they are automated.

It has been found in several systems that if a user supplied asynchronous interrupt is presented during the printing of a log-on error message, the monitor returns control to accept a new log-on attempt without advancing the counter set to record the number of tries. This permits automation of exhaustive enumeration of log-on's at maximum line speed without affecting the number of log-on's possible in a unit of time. It is also interesting to note that this attack can be executed with no indication that it is taking place. In general, the type of situation being sought in an asynchronous interrupt attack are operations in the command system that will cause control to be redirected to a location other than that had an asynchronous interrupt not taken place.

The problem with asynchronous interrupts is that the designers of the system chose to respond to the interrupt immediately rather than deferring interrupt response to a point in the program that permits a definitive determination of the state of the user program. While the effects cannot be predicted in advance, results with contemporary systems indicates that as an attack mechanism, it can bypass some security controls and is worth trying particularly if the high payoff attacks fail.

Trojan Horse*

This rather interesting attack is directed to placing code with trap doors into a target system. It attempts to achieve this by presenting the operators of the system with a program so useful that they will use it even though it may not have been produced under their control. An ideal 'gift' of this kind would be a text editor or other major system function that requires access to user files as part of the function. If the Trojan Horse routine opens the user files for him as part of the 'service', the program also has the opportunity to record the user ID and/or passwords on his file. It may also be possible to copy all or part of the file being 'edited' to a file accessible to a penetrator.

Details of exploiting this attack are highly dependent on the system on which it is to operate. In essence it bypasses any and all security controls that may otherwise

*This attack was identified by D. J. Edwards.

exist on most systems. It is the quintessence of the malicious threat against contemporary systems. To this extent, the Trojan Horse attack is directed against the procedural controls surrounding the use of a system. Such an attack can only succeed in environments where control over applications or other programs put on a system are lax. Unfortunately, this is the case for too many systems, even in environments where security appears to be of great importance.

Clandestine Code Change

A clandestine code change is related to the Trojan Horse attack in that it attempts to inject code that contains trap doors into the system for exploitation by a penetrant. Unlike the Trojan Horse, the clandestine code change is directed to placing ones own copies of crucial parts of the operating system into the system. The method used might be to send rigged system changes to the target system operators that appear legitimate. Obviously, if legitimate changes can be diverted and rigged, this can be used as well. In contemporary operating systems, opportunities for placing manipulated code are manifold because of the complexity of the system. It is only necessary for the clandestine code change to return control in supervisor state to the caller of an otherwise innocuous system function. The call can be 'keyed' by an arbitrary number set in one or more registers in order to minimize the possibility of accidental discovery. Once again what is being attacked is the procedural controls external to the system proper, and the fact that security controls are not isolated.

Asynchronous Attack*

The asynchronous attack attempts to exploit the independent I/O capability of modern computers by setting up conditions that cause the I/O to reference memory space that may be shared (with the user) by the monitor (see Implied Sharing above). This attack can take several forms. As an information recovery attack, the program can initiate a repeated (chained) output operation from an area in user assigned memory that is used by the monitor to store security sensitive information (e.g., the System Master Catalog entries used in file system operations in GCOS III). Because many systems will return control directly to the user program upon initiation of the I/O operation, (to permit parallel computing and I/O) it is then possible to call the system function that uses the shared space. With possibly some timing adjustments, the previously initiated output operation should be able to get a 'snapshot' of what the monitor places in that space, so even if the monitor zeros the space after it is through, the user will have copied it out onto a file for later examination.

Another form of this attack may give supervisory state control to the user if the monitor stores registers (including the instruction counter) in part of the memory assigned to a user, while the monitor is in control. This is a feature on some systems that is invoked when the monitor's primary register save area is filled up by nested intra-monitor calls. To mount the attack, the user constructs a record having

*This attack was identified by Major Roger Schell.

an address pointing to his code in the proper location. This record is replicated on a file and is then read by a repeated (chained) Input operation into the area of the user's memory used as overflow storage for registers. The user then makes a monitor call that causes the monitor to make a number of nested intra-monitor calls. Again, with the possibility of timing adjustments, the registers saved by the monitor will be overlaid by the input record containing the address pointing into the user's code.

Attack Scenario

Described below is an attack scenario developed against an HIS 635/GCOS III system that provides support for general program development in both time-sharing and batch modes.

The operators of the system recognized the potential vulnerability of the system to attacks involving unrestricted programming. As a consequence, the remote batch assembly language programming capability (CARDIN accepting the GEMAP assembly language) was removed from the capability of the general users who accessed the system from terminals in a time-shared mode. Programming in BASIC and in Time Shared (TS) FORTRAN was permitted to this class of users. Because the operators did not have the resources to certify the I/O operations in FORTRAN, or the possibilities of bypassing their controls in subroutines, both the I/O and subroutine capabilities were removed from the TS FORTRAN compiler as well. The BASIC System sequencing and control was found to be largely interpretive in nature, and was allowed to remain on the system unmodified except to remove a capability to make possible direct machine language patches to the generated BASIC code.

The problem facing a penetrator was first to find a means of breaking out of the FORTRAN or BASIC envelope to plant his own code, and then cause its execution. Secondly, with this 'programming' capability he then had to find and exploit a design or implementation flaw in GCOS III. Because of the interpretive nature of BASIC, it was decided to concentrate the attack efforts on FORTRAN.

Time Sharing (TS) FORTRAN Break

The essence of breaking out of TS FORTRAN on a HIS 635/GCOS III system was to discover a means of transferring control into data. Tests conducted on an HIS 635 at another site confirmed that the run-time package for TS FORTRAN checked array references at least at the main program level and that it was not possible to use that method on the target system. Because the subroutine and file capabilities were "removed" from the target system's FORTRAN, the methods involving overwriting an array with file data beyond the array boundary or spoofing the run-time package by referencing an array with negative or exaggerated indices from a subroutine were also effectively blocked. Investigation then centered on the Computed and Assigned GO TO statements. It was quickly ascertained that the Computed GO TO is checked to see that the switch variable is within the range of the label list. However, it was found that the Assigned GO TO was compiled as a direct transfer to the label specified and that the compiler did not distinguish an integer variable used for an Assigned GO TO

from an ordinary integer variable. It was also found that index register 3 was used by the compiler to hold the index value (offset from relative memory location zero) corresponding to a subscript in an array. Using these conditions it was a simple matter to construct a sequence that would cause a transfer to the first word of an (integer) array which was prefilled with instructions to be executed (using the DATA statement). The sequence is shown on the following page.

The reason this sequence permits one to "break out" of the TS FORTRAN envelope is because the compiler (and run-time package) does not distinguish between integer variables used in Assigned GO TO's and those used normally.

Recovery of User ID and Authenticators

The recovery of USER-ID and authenticators on the HIS 635 system was possible due to the existence of unrepaired deficiencies in GCOS III. It was discovered in an analysis of the GCOS DRL's and MME's that the buffer space made available by the caller to the File System (FILSYS) modules was not zeroed out before return to the caller. Based on this information the vulnerabilities were verified for the target system. In particular it was found possible to systematically scavenge the System Master Catalog (SMC) by presenting FILESYS with a catalog string of user "names" 0, 1, 2, ... In determining that these numbers were invalid catalog names the system had to read the portion of the SMC which would contain the false name in order to find that it was not present. Upon detecting this condition, FILESYS returned an error indication to the caller but did not clear out the buffer space before returning control. Upon regaining control the user merely read out the user-IDs and passwords from the buffer. This scavenge attack is insidious since it does not leave any trace of its activity and the user-IDs and passwords recovered permits its user to masquerade as any other user of the system.

The basic attack was developed by using the FORTRAN break (see above) to execute code placed in an array. The program used is shown on the following page.

This general method was tried on the target system using file activity function codes 3, 9, 5, and 21. Variations on the basic routine permitted printing the data in octal or an edited format.

There are a number of factors contributing to the success of this attack. These are:

a. The initial design flaw of using user-provided memory space for system buffer purposes.

b. The implementation flaw that does not zero out the buffer space before returning control to the user.

c. The operation flaw that the TS monitor does not deal harshly with users who supply a name that is not present in the SMC. This oversight is due to the fact that it is legal to present such a string (e.g., when adding a new file to the catalog) that must be checked for duplication.

```
10 DIMENSIØN INS(100)
20C
30C -----------------------------------------------------------------
40C INS IS THE ARRAY INTØ WHICH INSTRUCTIØNS ARE PLANTED
50C BY THE PENETRATØR
60C -----------------------------------------------------------------
70C
80 DATA INS(1)/Ø635004/
90 DATA INS(2)/Ø2755004/
100C
110C -----------------------------------------------------------------
120C THE VALUE ØF IBRK IS EQUIVALENT TØ TRA 0,3
130C -----------------------------------------------------------------
140C
150 DATA IBRK/Ø710013/
160C
170C -----------------------------------------------------------------
180C SETS UP THE RETURN
190C -----------------------------------------------------------------
200C
210 ASSIGN 200 TØ N1
220C
230C -----------------------------------------------------------------
240C PLACES THE RETURN IN THE ARRAY
250C -----------------------------------------------------------------
260C
270 INS(3)=N1
280C
290C -----------------------------------------------------------------
300C ASSIGNS VALUE 1 TØ INTEGER VARIABLE N2
310C-----------------------------------------------------------------
320C
330 N2=1
340C
350C -----------------------------------------------------------------
360C NEXT STATEMENT CAUSES X3 TØ BE LØADED WITH THE ADDRESS
370C ØF THE FIRST WØRD ØF THE ARRAY INS
380C -----------------------------------------------------------------
390C
400 INS(N2)=INS(1)
410C
420C -----------------------------------------------------------------
430C CØMPILES AS A DIRECT TRANSFER TØ THE INTEGER VARIABLE IBRK
440C -----------------------------------------------------------------
450C
460 GØ TØ IBRK
470C
480C -----------------------------------------------------------------
490C CØNTRØL RETURNS HERE FRØM CØDE IN INS
500C -----------------------------------------------------------------
510C
520 200 PRINT 201,INS(4)
530 201 FØRMAT(1X,Ø12)
540 STØP
550 END
```

```
20 DIMENSIØN IFIL(33)
30 DIMENSIØN ICHR(64)
40 DIMENSIØN IDUM(13),JDUM(24)
50 DIMENSIØN IBIG(24)
60 ASCII ICHR,JDUM
70 DATA ICHR/Ø060000000000,Ø061000000000,Ø062000000000,
80& Ø063000000000,Ø064000000000,Ø065000000000,Ø066000000000,
90& Ø067000000000,Ø070000000000,Ø071000000000,Ø043000000000,
100& Ø043000000000,
110& Ø100000000000,Ø072000000000,Ø076000000000,Ø077000000000,
120& Ø040000000000,Ø101000000000,Ø102000000000,Ø103000000000,
130& Ø104000000000,Ø105000000000,Ø106000000000,Ø107000000000,
140& Ø110000000000,Ø111000000000,Ø046000000000,Ø056000000000,
150& Ø135000000000,Ø050000000000,Ø074000000000,Ø134000000000,
160& Ø136000000000,Ø112000000000,Ø113000000000,Ø114000000000,
170& Ø115000000000,Ø116000000000,Ø117000000000,Ø120000000000,
180& Ø121000000000,Ø122000000000,Ø055000000000,Ø044000000000,
190& Ø052000000000,Ø051000000000,Ø073000000000,Ø047000000000,
200& Ø053000000000,Ø057000000000,Ø123000000000,Ø124000000000,
210& Ø125000000000,Ø126000000000,Ø127000000000,Ø130000000000,
220& Ø131000000000,Ø132000000000,Ø137000000000,Ø054000000000,
230& Ø045000000000,Ø075000000000,Ø042000000000,Ø041000000000/
240 DATA IFIL/Ø36002000,Ø001356,Ø2001411,0,0,0,0,
250& Ø001361000000,Ø001372001363,Ø001364000000,0,0,
260& Ø740000000000,0,Ø2000002,Ø510102010000,Ø000220202020,
270& Ø760000000000,Ø777777777777,Ø510102010000,
280& Ø000220202020,Ø202020202020,Ø202020202020,Ø777777777777,
290& Ø510102010000,Ø000220202020,Ø202020202020,Ø202020202020,
300& Ø102122113062,Ø040040040040,Ø040040040040,Ø040040040040,
310& Ø777777777777/
320 KKK=2
330 LLL=2
340 DATA IP3/Ø1373/
350 DATA IP4/Ø1376000000/
360 DATA IP5/Ø1414001400/
370 DATA IP6/Ø1401000000/
380 DATA IBRK/Ø710013/
390 DATA IP77/Ø770000000000/
400 DATA IP1/Ø2001433/
410 DATA INS(1)/Ø635004/
420 DATA INS(2)/Ø2755004/
422 DATA IP98/Ø060000000000/
430 DATA IP99/Ø51010303000301/
440 GØ TØ(600,601),KKK
450 600 CØNTINUE
460 ASSIGN 677 TØ N1
470 INS(3)=N1
480 N2=1
490 INS(N2)=INS(1)
500 GØ TØ IBRK
510 677 PRINT 678,INS(4)
520 678 FØRMAT(1X,Ø12)
530 GØ TØ 602
540 601 CØNTINUE
550 DØ 77 I=1,33
560 INS(I)=IFIL(I)
570 77 CØNTINUE
580 INS(3)=IP1
590 INS(2)=IP3
600 INS(8)=IP4
```

```
INS(25)=0
INS(16)=IP99
INS(26)=0
INS(26)=0
101 FØRMAT(1X,I4,1X,Ø12)
66 CØNTINUE
PRINT 101,INS(25),INS(25)
DØ 79 I=34,500
INS(I)=0
79 CØNTINUE
ASSIGN 200 TØ N1
INS(4)=N1
N2=1
INS(N2)=INS(1)
GØ TØ IBRK
200 CØNTINUE
GØ TØ(604,605),LLL
604 CØNTINUE
DØ 615 I=1,40
IF(INS(I)) 616,615,616
616 PRINT 101,I,INS(I)
615 CØNTINUE
GØ TØ 602
605 CØNTINUE
J=0
GØ TØ 714
715 CØNTINUE
IF(INS(102+J) .NE. INS(105+J)) GØ TØ 805
IF(INS(102+J) .EQ. 0) GØ TØ 805
714 CØNTINUE
IF(INS(102+J) .EQ. IP77) GØ TØ 716
INS(102)=INS(105+J)
INS(103)=INS(106+J)
INS(110)=INS(110+J)
INS(111)=INS(111+J)
5 FØRMAT(1H ,3(Ø12,1X))
ENCØDE(IDUM,1)INS(102),INS(103),INS(110),INS(111)
1 FØRMAT(4(Ø12))
DECØDE(IDUM,13)(IBIG(I),I=1,24)
13 FØRMAT(48(Ø2))
DØ 99 I=1,24
IK=IBIG(I)+1
JDUM(I)=ICHR(IK)
99 CØNTINUE
PRINT 3,(JDUM(I),I=1,24)
3 FØRMAT(1H ,12(A1),1X,12(A1))
713 J=J+12
GØ TØ 715
716 CØNTINUE
INS(25)=INS(25)+1
GØ TØ 66
805 J=J+1
IF(J .GT. 500) GØ TØ 716
IF(J+100 .GT. 500) GØ TØ 716
IF(INS(100+J) .EQ. IP98) GØ TØ 807
IF(INS(100+J) .EQ. IP77) GØ TØ 716
GØ TØ 805
807 CØNTINUE
GØ TØ 715
602 CØNTINUE
```

While there are other vulnerabilities of the HIS 635 that could have been used, the specific method outlined above is typical of a security penetration of contemporary systems. Although this specific attack was mounted against the system from a terminal through the Time Shared monitor, other avenues are possible from batch programs as well.

Summary

A number of areas of weakness of contemporary systems have been outlined above. The availability of listings of the operating system will speed the penetrator's efforts many times over, although even without such aid, a systematic probing of operating system function calls, followed by dumps immediately after the call would produce similar results.

A major point is that with no recognized principles of design for security, the ad hoc protection mechanisms of most contemporary systems are insufficient to defend against a dedicated penetrator.

APPENDIX II

A SURVEY OF THE STATE-OF-THE-ART OF COMPUTER SECURITY TECHNOLOGY

INTRODUCTION

This appendix is an assessment of the current state of the art in computer security technology. It is an attempt to put the technical problems into perspective and to identify what appear to be outstanding problems, and what additional work is needed to solve them.

ACCESS CONTROL TECHNIQUES

This area comprises two categories:

a) Control of access to a system.

b) Control of access to the elements of system (hardware and software).

Basically, the first category involves authenticating users to the system. For remote access users, two techniques are available — control of physical access to a terminal, and use of 'passwords' as authenticators. The former technique is an instance of physical security that will not be dealt with further here.

Authenticator schemes such as automatic fingerprint reading, the Identimat hand geometry reader, and read/write magnetic card readers appeal to the gadget minded, but offer no additional security over the password schemes. The magnetic card reader-writer may be a more convenient medium for one-time passwords, but is limited in application (as is the scheme outlined below) to situations where the communications lines are protected.

The technique of using passwords to authenticate a user to resource sharing computer system is well known. Almost all of the systems in use in Government, and all of the commercial time-sharing systems use this technique. In Government systems, the password is classified, and considered 'secret' (not the national classification) because the password is equated to the combination of a safe containing classified material. This analogy is incorrect, since the access to material in a resource sharing system is in nearly all cases controlled by the user's identifier (generally not a classified item). The password in these systems plays the role of an authenticator, that is verifying to the system that the user is who he claims to be. The authentication takes place by the user supplying his unique password along with his identifier. Since the password is presumed to be known only to that user, the presentation of the password uniquely associated with a given user identifier is taken as prima facie evidence that the user is indeed who he claims to be.

While the password does not give access to material in the system (the user's identifier does), it does give a user access to the system. To this extent, the analogy to a combination is correct, however a better analogy is that the password is equivalent to the combination to a vault containing a variety of safes each of which contains classified material. Once inside the vault, the user's identifier will open his safe but none of the others.

There are basically two reasons that the combination to a safe containing classified information must be classified to the level of the material in the safe and be considered 'secret'. The first is that it is not economical to change the combination of the safe after every use and even if it were, the problems of distributing the changed combinations even to two or three people who might share the safe are overwhelming. Obviously if only one person uses the safe, repeated changes are unnecessary except for presumptions of carelessness by the owner. The second reason for considering safe combinations 'secret', particularly in the single-user case is the inability to detect when a safe was opened using the combination. For these reasons, the safe owner and user(s) are prohibited from writing down the combination.

The principal reason the authenticating password is kept secret is that, like safe combinations, it is reused for extended periods of time. Further, if it is observed (like safe combinations), it would permit another person to masquerade as the legitimate user. Because safes are most often located in the immediate vicinity of their owner/user, they cannot ordinarily be entered unobserved. A masquerader, however, could enter a resource sharing system from another terminal, unobserved by the affected legitimate user. Depending on the length of the password period*, a masquerader could effect a long-term penetration of another user's files with a low probability of detection. This is a serious risk resulting from use of long-term passwords.

Requirements for a One-Time Password System

It is to counter the risk of long term penetration that one-time password schemes have been proposed. Weissman in the design of Adept-50 provides for a table of up to 64 passwords that can be used to implement a one-time password scheme. However, even his scheme considers the passwords 'secret.'

The major drawback to one-time password schemes is the cost and difficulty of distributing lists of passwords to a large number of users, particularly in situations where the rate of password use varies widely over the user population. To be most effective, the one-time password system should minimize the distribution problem.

'Secret' passwords tend to be long because they must preclude exhaustive testing for the duration of the password period. For a one-time password system, shorter

*The period of time a password remains in effect unchanged.

passwords can be used provided they are long enough to provide adequate variability over the set of users of a system. If the log-on procedure is designed to permit only three incorrect log-on attempts before locking out a user from subsequent log-on attempts, then random sequences of 3 or 4 letters could be effective, providing 17, 576 or 416, 976 possible passwords to 'cover' a set of users.

A Centrally Distributed One-Time Password Scheme

As part of the log-on sequence, it is possible for the system to generate and return to the user a new random sequence password for use the next time he logs-on. Before transmitting the new password, the system can check the list of current passwords to eliminate current duplicates (although it is not clear that with one-time passwords duplicates arising at random constitute a lessening of the security feature provided by one-time passwords).

Because the password is good only for the next log-on, it could be printed (without further identification) and retained in the possession of the user without special security controls.

A one-time system such as this requires a way of giving a new user his first password in a controlled way. For this purpose and for any subsequent case (see discussion below) where the user's current password cannot be used, a special program, available for execution only by a System Security Officer (SSO) (from a terminal) would be used. The program would be called by the SSO (after logging-in with his current one-time password) and would accept as input the user-number (user-id) and optionally the access permissions to be assigned to him. After entering the new user in the list of authorized users, the system would generate and return an initial password for that user. To keep the password private, the SSO could remove himself from the terminal while the new user received his initial password.

The principal risk with either form of passwords is compromise of the password by exposure to unauthorized persons. The effects of such compromise are considerably different depending on which password scheme is used. For the open one-time password scheme, the effects range from none (in the case the legitimate user logs-on again before the masquerader can use a surreptitiously obtained password) to being denied access to data at a crticial time. For the 'secret' passwords, the effects range from none (under the unlikely case of verification of user activity as part of an audit procedure) to a long-term exploitation of the user's data base. The open password scheme is vulnerable to denying a user access to the system if the password is compromised and used by a penetrator.

Summary

A comparative analysis of the vulnerabilities of one-time passwords (centrally distributed) and 'secret' passwords favors the scheme of one-time passwords authenticating users to resource-sharing systems. It appears that the risks of delivering the one-time password in open hard copy and permitting its unrestricted retention by the

user are virtually nil, and in any case are significantly less than the commonly accepted 'secret' password schemes currently in use.

The main objection to passwords as authenticators is the distribution problem, which for systems of any size becomes so costly that the password is used for extended periods of time. This increases the risk that a surreptitiously observed password can be exploited by the observer for corresponding periods. The one-time centrally distributed password scheme described above is suitable for use where the communications are adequately protected.

This aspect of the computer security problem is well understood and manageable with present technology.

HARDWARE

The current state of computer hardware varies considerably. In spite of this, virtually all of the so-called 3rd generation computers have the essential elements for constructing penetration proof operating environments for limited use. These elements are the two-state operation and hardware storage protection. Virtually <u>none</u> of these systems have devised check circuitry that assures the proper operation of these elements. Limited experience with a few time-shared systems indicates that hardware failures that suspend storage protection or permit user-mode programs to execute master (supervisor) mode instructions are infrequent indeed.

There is growing evidence that descriptor-driven machines provide an excellent base for constructing penetration proof operating systems, although it is clear that even with such aids it is necessary to exercise care in the implementation of the operating system.

The appeal of descriptor-driven machines is severalfold. First, it provides an environment that encourages building the operating system in a structured way. Second, it is possible to separate addressability from privilege, making it possible to operate virtually all of the operating system in a non-privileged mode. With all storage references interpreted by descriptors, it is possible to more effectively apply selective permissions (read, write, execute, etc.) to different parts of the operating system. Third, the portion of the operating system dealing with real resources (memory, peripherals, file space, etc.) can be localized and made as secure as need be for securing the system. Finally, descriptor-driven (virtual) machines make it possible to include the operating system in the user's address space in a protected way, thus facilitating intra-process communication, and enforcing separately the controls for reading (data or programs), writing and execution.

It is interesting to note that penetration attacks on conventional two-state, non-descriptor machines are generally directed to obtain supervisory state control of a system. This in turn permits the successful penetrator to manipulate operating system code at will. Because global addressing in contemporary systems in linked to the supervisor state of such systems, it is often necessary to enter this state merely to

provide addressing facilities to part of the operating system. With so much of the operating system having to be in supervisor state for addressing reasons alone, it makes it extremely difficult to avoid exploitable implementation flaws in the operating system.

The increased use of microprogramming to implement instruction sets in contemporary systems makes it feasible to incorporate part of the operating system code in microstore. However, the tables representing active processes must still reside in regular memory and are subject to potential manipulation. The principal benefit from putting parts of the operating system in microcode is the increased attention it will get during design and implementation rather than any special security properties of microprogramming. For special applications, it is feasible to microprogram higher level language interpreters thus removing users further from the real machines.

Summary

By and large, the current state of hardware development will support the design and implementation of penetration-proof operating environments. The descriptor-driven virtual machines make this process simpler because of the ability to specify and control reference permissions separately from the privileged state of operations and the localization of real resource (memory, files, etc.) inventory management.

COMMUNICATIONS

While current technology provides good techniques for secure communications, there is still no communications security (COMSEC) equipment available designed specifically for interactive terminal to computer connections. Further, the requirement to physically protect crypto gear (vaulting it, for example) makes the cost of applying this technology quite high indeed (perhaps $35, 000-$50, 000 per vault). Compounding this is the problem of distributing keying materials for several hundred or thousand terminals in a system.

Current activity in this area is promising. ARPA is pressing for secure terminals for the ARPA network. The USAF has under development "office environment" secure terminals for the LDMX program, and there are techniques available for remote keying. All of these efforts are at least 3-6 years away from being able to deliver useable equipment for general use. Even so, all of these efforts are directed to secure communications from reading by unauthorized persons. In at least one environment (AF/ACS) there is an immediate requirement to secure unclassified communications against intrusion that appears more as a write protect problem than read protect. The main implication of this requirement is that extraordinary protection of the anti-intrusion equipment is not necessary because of the unconcern for the content of the traffic being transmitted.

Given the inevitable development of suitable low cost COMSEC equipment for interactive terminals, the burden then falls at the computer site to find room for crypto equipment at the computer side of the links. In order to reduce the space,

power and air conditioning requirements at a site, it will be necessary to develop time-shared COMSEC equipment that could even be stored programmed (microprogrammed) and dedicated to this function. The equipment could either be viewed as an integral extension to the presently available communications front end processors, or as a separate function interposed in front of a communications processor.

Summary

The technology and techniques applicable to these problems are available with little or no additional research. There are no COMSEC products available designed specifically for use with remote interactive terminals in an "office environment." Development currently underway will refine the techniques, but do little for the current problems, even when they become available, unless these problems are specifically addressed in a development program.

FILE SYSTEMS

Basic Problems

There are basically two file protection problems (excluding the problem of physical protection). The first arises in connection with computer utilities, and is concerned with methods of precisely controlling the sharing of information, and more specifically programs. The problem is complicated by the notion of implied sharing. As an example, if a user B is sharing some programs owned by user A, and then authorizes user C to share his program that in turn shares some of user A's programs, how is the sharing between B and C controlled such that C does not have access to the programs of A and B but only to their results. Basically, the question being addressed is how can communication be established between two users' programs such that only the results of the shared program are available to the sharer.

The second problem arises in environments where data is classified according to external criteria (e.g. in files of defense information), and is more concerned with establishing a logically consistent method of determining the security label to be associated with file access requests in order to permit an intelligent determination of the validity of the request. This problem is complicated by the fact that users, programs, terminals, files, and executions all can be considered to have such labels, and that the security label of some objects (executions and some files) can change during the execution of a program, or during the execution of a job. In addition, in the environments where this problem is important there is considerable attention paid to the derivation and proper transfer of security labels to files and printed material.

Models for Shared Information Processing

The issues involved in this problem are how authorization to use a file or a program is accomplished and how the general framework in which programs are created and executed.

Most of the workers involved with this problem have assumed or required the existence of a file system consisting of a collection of files, and a directory associating a user with his files, or in exceptional cases a directory associating a file with its users. Assuming the first form, the authorization mechanism must permit a file owner to designate the users with whom he wishes to share a file, and those privileges the sharer is permitted with respect to the file. A commonly used mechanism is to associate with each shared file in a user's directory, a list of other users who may access the file, and for what purpose (i. e., read, write, append, etc.). A sharer, in order to establish a connection to the shared file creates his name for the file, and equates it to the file being shared. Sharers reference to the file name he created is interpreted as an indirect reference to the owner's directory, from which the type(s) of access permitted are checked before completing the reference. A number of variants on this scheme can occur to make the process more efficient. For example, the directory search can take place at binding time (assuming pre-execution binding), a name substitution made, and a transfer of access flags made to the sharers' file control block. However, these are implementation and application dependent, and will not be discussed further here. In one model[1], actual system commands are provided to permit designating sharers of files.

Other authorization models exist; these include use of passwords associated with each file in the (protected part of the) system to act as locks. An owner authorizes sharing of his files(s) by providing the sharer with the password for the file. As Friedman[1] notes, however, this is less than satisfactory because it permits the sharer unrestricted access to the file for any purpose.

The method of actually controlling authorized sharing in nearly all utility-oriented systems is based on the use of indirect references to the shared objects through descriptors. It is characteristic of most systems designed for information utilities, or large populations of on-line users that they provide some form of virtual memory system.[2] The objects (e. g. programs, data, files) occupying the virtual memory are represented by descriptors, collected into one place, managed by the system, and acting to map a virtual address into a real address. The mapping is often aided by hardware in the system, but this is merely a technique for improving execution efficiency, and is not fundamental to the concept.

Since descriptors are maintained by the system (necessarily, since they deal with real resources) they are in a special segment designated READ-ONLY to a process.

Descriptors are used to control sharing in a variety of ways. Basically, each descriptor, representing a program, data set, file, etc., contains control information in addition to the address in real memory where the object is located. The basic control information of security interest is the type of access permitted to the object — READ, READ-WRITE, EXECUTE, APPEND, etc. Since the operating system is the only program permitted to create and manipulate these descriptors, the necessary mechanism to provide controlled sharing of other users' programs and files appears to be established.

This would be the case if only one user at a time were permitted to gain access to an object. However, in the multiple user environment, a given object could be in use by a large number of users, perhaps with different access privileges. In general, this case is handled within the same framework as for the single user; since each user's process is represented by a descriptor table (segment) unique to that user, the descriptor referring to such an object can have the access control information set to the appropriate value for that user. The actual checking on access type is accomplished on modern systems in hardware as a descriptor is referenced. Within this general framework, a number of secondary problems emerge. Graham[3] treats protection as a disjoint series of rings, and discusses the problems of changing control from one protection level (viewed as concentric circles or rings) to another in a safe manner. To provide protection in both a downward (from a superior routine to an inferior routine) as well as an upward direction, he proposes a model that augments the descriptor for a segment with ring bounds that permits free access as long as the element being transferred to is within the bounds but invokes special software whenever the bounds are exceeded in either direction. In general, the special software validates the address being referred to regardless of the direction of the reference. In this way, the mechanism protects a process from the operating system as much as the other way around.

Vanderbilt[4] has created a model that extends that of Graham to include cases that arise when a user sharing an object authorizes others to use the process he creates. In his model, he introduces the notion of access privileges as a function of the activation level of the process, and in effect makes copies of the descriptor segment for each activation level encountered in order to provide the precise control needed. He distinguishes the problems that arise from direct access to a shared procedure, and adopts as part of the model the policy that direct sharing of procedures is only permitted for procedures authorized to the borrower by their owner, while only indirect sharing of procedures is permitted for those procedures owned by a third party and authorized and used by an owner in constructing a procedure that is (to be) shared with others. In the latter case, a borrower can only affect indirect access to procedures borrowed by the owner of a shared procedure.

Models for Hierarchical Access Control

The only available work that deals with this subject in a formal manner is that of Weissman[5]. In it the author defines security objects (files, users, terminals, and jobs) and security properties associated with the objects. The properties are Authority (a hierarchical set of security jurisdictions — classification), Categories (a mutually exclusive set of security jurisdictions — a formalism of the need-to-know policy), and Franchise (clearance).

The balance of the paper is devoted to developing a set-theoretic statement of the policy adopted in the ADEPT-50 system:

a) A user is granted access to the system only if he is a member of the set of users known to the system.

b) A user is granted access to a terminal, only if he is cleared to do so.

c) The clearance of a job is determined from the clearance of the terminal and the clearance of the user.

d) Access is granted to a file if the clearance and need-to-know properties of the file, and the user is authorized (cleared) to the job.

The model treats all file accesses as events, and maintains a running determination of the classification and need-to-know level of the job based on events throughout its execution. This information, known as a high water mark, is most useful in determining the derived classification and need-to-know for new files created during job execution, and for labeling output.

The only drawbacks with this model is that classification and need-to-know can change in only one direction — upward (to higher levels), depending on the files used in the application. Two relatively infrequent, but none the less important cases are not treated by the model — the case where individual data items are themselves not classified, or are a low level classification but when aggregated (collected into a file or report) may acquire a higher classification, and the case where a program transforms a classified file into an unclassified file (perhaps by extracting data known to be unclassified for a report).

The latter case arises principally because the classification is applied to too large a unit (the file), and would disappear if fields could be individually classified. The former case cannot be handled within the framework of Weissman's model as it stands, since it is a value judgement as to when (or if) a particular aggregation requires a higher classification than the source material. This could be handled by providing the concept of security declarations in programs that would override the running classification and need-to-know property if specific conditions were encountered during execution of the job. The conditions might be of the form, 'If the number of records placed in temporary file F1 is greater than 100, advance the classification to the next-highest level', or in general IF <condition> THEN <statement of security labeling>.

File encryption techniques are available that will provide virtually any degree of protection desired. High grade algorithms can be operated at a cost of 60 to 100us per word enciphered or deciphered. Because of the problems with contemporary operating systems, key protection cannot be assured restricting the technique's principle value to media protection. In descriptor based systems, where key protection could be better assured, the technique could also be used to provide additional protection of files. The media protection problem solved by this technique should not be underestimated, since media 'contaminated' with classified material becomes difficult to dispose of.

Summary

Really advanced file systems with arbitrary sharability are not common. A number of models for building such systems exist, but only limited experience has been gained with them. The bulk of file models are not designed with government security classifications in mind. File encryption, while feasible is still too costly in execution time for widespread use.

SECURITY SURVEILLANCE AND AUDIT TRAILS

As presently conceived, security audit trails are of little value in detecting unauthorized activity, either because they do not contain sufficiently useful information, or worse because the data is not examined by security personnel.

The entire concept of taking fixed content 'snapshots' of each user's activities is wrong primarily because it doesn't give enough of the right kind of information in cases of interest, while giving too much information in the bulk of the cases.

The emphasis on an audit capability is a reflection of the desire to conduct security surveillance operations in a resource sharing system in order to detect breaches of security or penetration attempts.

Unfortunately, the audit schemes developed around existing facilities (mostly accounting oriented) in contemporary systems are too inflexible to provide either surveillance or a damage assessment capability to systems security personnel.

The audit capability is related to the instrumentation of a system. To date the emphasis on (hard or soft) instrumentation has been for system performance measurement. While it can be seen that a security audit capability requires many of the same points of measurement, the security audit differs in what is recorded, and more importantly how it relates the measurement to the real world of users, terminals, communications lines, etc. Further, from a security audit viewpoint, while <u>all</u> possible measurements are not of interest all of the time, all possible measurements will be of interest (not all at once) at some time. Further, the systems must be capable of supporting a variety of security surveillance audits at different levels of detail simultaneously. For example, it must be possible to monitor (record) each direct and induced transaction on behalf of one or more specific users, while maintaining a running record of the use of several of the communications links, while recording all transactions (by each user) against the files on a particular physical storage device, and to be able to vary the mix and focus easily on a day-to-day or shorter time basis.

Yet be determined are the most promising way of relating a user, terminal, physical device, etc. to the measurement points, and how to vary as a function of the level of surveillance being maintained, what is recorded upon reaching a given (program) measuring point. While it seems reasonably clear that both hardware and programs can be provided measuring points at little cost, the best way (or even

alternate ways) to achieve the desired security audit capability is not yet well understood.

Summary

Instrumentation (of software) is relatively new, but appears to offer no particular technological problems in the usual case. The ability to survey an aribtrary and changing mix of users, terminals, files, etc. in a fully instrumented system involves being able to relate representations of what is under surveillance to measuring points in a system. There do not appear to have been any efforts in this direction to date.

OPERATING SYSTEMS AND SOFTWARE

It is in this area that the major problems of computer security arise. These problems are those related to

a) incorporating security requirements into software specifications,

b) the scope of the security problem (i. e. system-wide),

c) the primitive state of technology regarding 'proof' of correctness of programs,

d) the lack of a definition of 'correctness' as it impacts computer security.

The problem can be considered in various ways. Even with a collection of individually 'correct' programs, it may be possible to attack the system by exploiting design omissions or flaws. It is hypothesized that exploitable penetration attacks on computer systems are possible because the operating system contains either of the implementation or design flaws listed below.

a) Placing or making available system state information in user's address space (the concept of user's address space must be expanded to include implied resources associated with a program such as job files, swap files, etc.)

b) Providing too big an addressing context for 'normal' systems functions.

Instances (from GCOS III) of the first item include the placement of a program's Slave Service Area (SSA) onto the checkpoint file with the program being checkpointed, using the slave prefix area for register storage from supervisory functions, placing a copy of the SSA on the *J file after peripheral allocation, and the use of buffer space from a slave program for catalog searching functions.

The second item is a function of the hardware design, and is sometimes seen as incomplete address parameter validation. An instance from GCOS III involves the ability to move a User Status Table (UST) (the 'state' of a time-sharing user) to anyplace within the time-shared subsystem.

Even if the hypothesis is true, and avoiding the two classes of flaws described above is a sufficient condition to obtain a penetration-proof system, the statement of the condition is too broad, and is equal to saying 'don't make any mistakes.' In effect we are saying that we have no useful models of a penetration-proof operating system (environment) against which to measure proposed or actual implementations.

Another aspect of this problem is that it is clear that only some parts of an operating environment need to be good (in some sense) to provide a penetration-proof system. To take an absurd example, no one would consider attacking a system by attempting to manipulate a sine routine, or a random number generator, or a sorting algorithm. However, the areas of an operating system that need to be protected are not cataloged and are not available for evaluation.

In general, we are concerned with the problem of preventing a malicious user from seizing control of a system, or exploiting design or implementation flaws to gain unauthorized access to data. The capability models of Lampson[6], elaborated by Denning[7] appear to provide a basis for identifying principles upon which secure systems can be built.

The structured programming techniques of Dijkstra appear to offer a good model for how systems should be constructed. Their principal value seems to be in being able to comprehend the result, something not easily done now. Even if the technique provides 'proof' of algorithm correctness, it does not appear to offer proof that a design is complete.

Summary

Lack of good ideals in the form of a non-implementation model hinder discussing the security aspects of software. A number of promising techniques for constructing programs about which assertions can be made exist, but to date have not been applied to the issue of computer security. First priority should go to a penetration-proof systems model.

REFERENCES

1. Friedman, T. D., The Authorization Problem in Shared Files, IBM Systems Journal (9), 4 pp. 258-280 (1970).

2. Denning, P. J., Virtual Memory, Computing Surveys 2 No. 3, 153-189 (September 1970).

3. Grahm, R. M., Protection in Information Processing Utility, Communications of the ACM 11, No. 5, 365-369 (May 1968).

4. Vanderbilt, D. H., Controlled Information Sharing in a Computer Utility, MAC TR-67, Project MAC, Mass. Institute of Technology, Cambridge, Mass., 24 October 1969.

5. Weissman, C., Security Controls in the ADEPT-50 Time-Sharing System, Proceedings 1969 FJCC, pp. 119-133.

6. Lampson, B. W., Dynamic Protection Structures, Proceedings 1969 FJCC, p. 27.

7. Denning, P. J., Third Generation Computer Systems, Computing Surveys Vol. 3 No. 4, December 1971.

APPENDIX III

SECURITY ASPECTS OF DATA MANAGEMENT SYSTEMS

STUDY OF UNIQUE SECURITY ASPECTS

Because data management systems (DMS) are important and distinct elements of most command and control systems, management systems, intelligence systems, and logistics systems, a study was made of those aspects of DMS that are unique in relation to other types of computer-based systems and which contribute to security problems. A DMS provides facilities for the management of data - i. e., creation of a data base, update and maintenance, retrieval, and rearrangement of retrieved data - usually for data that is shared by several applications and users. The unique security aspects of DMS arise from the sharing of data by several users with different access authorizations (clearances, need to know, etc.) and from the large number of data elements that may be stored in a DMS (millions and possibly billions). Four major aspects of DMS which impact security are discussed in the following paragraphs.

DATA IDENTIFICATION

Control of access to data in a data base is dependent on identification of the data accessed and the security sensitivity of that data. Identification of data in an access request can be categorized in the following ways which are pertinent to access control:

- logical identification of the data accessed with respect to the data structure of the data base,
- physical identification of the storage location of the data accessed,
- direct identification of the data accessed or of its storage location,
- relative identification of the data with respect to some other data in the data base, such as the data accessed in the immediately preceding request.

An example of logical direct identification is "the length of the runways at airfield Norton". In this expression, "airfield" identifies a file of data concerning airfields; "Norton" identifies a record in the file airfield which contains data on the airfield at Norton Air Force Base; "runways" identifies a group within record Norton that contains data on the runways at Norton airfield; "length" identifies the numerical values of the runways. This example shows the naming of individual elements in a data structure and their aggregation into structural elements (file, record, group, etc.). Another example of logical direct identification is "data element 37 in data list 15".

Examples of physical direct identification are "the data in storage locations 3123 to 3146" and "physical record 12".

An example of logical relative identification as "the third data element in the data list following the element previously accessed".

Examples of physical relative identification are "the data element which is located 15 storage address units from the data element previously accessed" and "backspace 24 storage address units from the data element previously accessed".

It is readily apparent that relative identification creates problems for access control. If a DMS user accesses first a structual element (aggregation of data elements) which he is authorized to access, he could then specify by relative identification a structural element to which he is not authorized access.

For access control, each named structural element in a data base may be assigned a sensitivity parameter which specifies the access rule that is applied to control access to that structural element. Then when a structural element is referenced in a program the sensitivity parameter may be examined and the access rule specified by the parameter value for that element can be applied before the data in the structural element is made available to the program. A complication arises when the structural element name identifies a function or a relation - i. e., a rule by which the members of one set (or several sets) are assigned to members of another set - rather than a specific aggregation of data elements. Then the sensitivity (specification of the pertinent access rule) must be dependent not only on the name of the function (or relation) but also on the names of the data elements (or sets or subsets) to which the rule is to be applied.

In some cases, a security sensitivity may be associated with a device rather than data; e. g., data having a particular sensitivity may be stored only in a specified area of storage and data of other sensitivities may be stored elsewhere.

DATA DIFFERENTIATION

Within a DMS, data may be differentiated on several bases, which may affect the access control rules that apply to it:

- Agent - i. e., whether the data can be accessed directly by people, by a process, by a network node, or by some specified combination of them.
- Form - i. e., whether the data has the format of particular strings of characters - e. g., binary string, decimal floating point numeral, alphanumeric string, text, fixed length records, etc.
- Media - i. e., whether the data is stored on-line in core, disk unit, tape unit, etc. or off-line in cards, paper tape, disk pack, tape reel, image media, etc.
- Context - i. e., the denotation of the data and hence its sensitivity may be dependent on the context in which it is used (the process invoked to use it).
- Capabilities and permissions relating to its use - i. e., whether it can be read, written, or executed or whether the user can use the data only or whether he can create it, update it, replace it, or delegate access to it.

PROCESS DIFFERENTIATION

Access control may be affected by the processes involved in the DMS. They are discussed in the following paragraphs from three points of view.

Complexity

A DMS may have a simple structure - e. g., a single file or fixed length records - or a complex structure - e. g., multiple files with correlations both within files and between files. The more complex the structure the more difficult is access control. The more different kinds of structural elements there are in a DMS the more difficult it is to assure that access to all of them is controlled.

The amount of data stored in a data base under control of a DMS may be very large. Some data bases in active use have hundreds of millions of data elements (billions of characters) and data bases of billions of data elements may be expected in the near future. With such a large number of data elements, it is difficult to assure that the correct sensitivity has been assigned to each one; in fact, it may be nearly impossible to verify that the correct sensitivities have been assigned in such large data bases.

A large data base having a large number of users with differing access authorizations will have a substantial number of different access rules. This creates a complex situation with possibilities for error. The access rules must be verified to be correct and their assignment to structural elements must also be verified.

DMS - Human Interface

The interface between the user and the DMS can be classified into two types:

- Open - host language preprocessor. The user accesses the data base through a program written in a conventional programming language (host language) such as COBOL, FORTRAN, or JOVIAL.
- Closed - interpretive query language. The user accesses the data base only through a query language that is interpretively executed.

Examples of open-host language DMS are IMS, IDS, and DM-1. They have been designed to establish and maintain a data base so that it is accessible by batch programs. The DMS provides a data language which is used as an extension to the host language in the application program, which is compiled before execution. The user has at his disposal all the facilities of the host language and the data language, which he can use to try to break the access control of the system.

Examples of query language DMS are GIM and TDMS. In such systems, the user can access the data base only through the query language of the DMS, which has limited expressive power and is interpretively executed. Thus the queries are all under control of the system and their processing can be integrated with the access

control mechanism. A problem arises in that in the evolution of such systems pressure arises from users and potential customers to extend the capabilities of the system, such as, e. g., to add the capability to access the system from host language programs. A version of GIM has such an added capability.

DMS - OS Interface

A DMS may manage the machine resources it uses directly or it may manage them indirectly making use of the file management facilities of the operating system. In the first case, the access control is effected entirely within the DMS; in the second, access control may be divided between the DMS and the operating system.

Most operating systems provide facilities for the management of files; e. g., OS 360 provides various "access methods", such as ISAM, BTAM, etc., and EXEC 8 provides a Fastrand handler, symbionts, and utilities that manage the creation, maintenance, and use of "element files".

IBM's CP-67 provides an environment in which each user is assigned a virtual machine. Within his virtual machine, a user can deal directly with his virtual machine resources but he cannot use any resources outside of his virtual machine. The operating system interprets virtual resources into real resources and keeps users separate. In such a system, a DMS would function as though it were on a dedicated computer. IBM's recent announcement of the 370 virtual memory introduces machines in which program references to virtual storage are interpreted by a dynamic address translation facility.

The MULTICS system provides an environment in which the user interacts with an abstract machine in which all storage is treated homogeneously. Stored information is compartmentalized into "segments". A user accesses data by naming the segment and the address of the data within the segment. Each segment has a set of access attributes that specify the way in which a user is permitted to reference the data in the segment.

DATA AGGREGATION AND INFERENCE

Control of access to individual data elements and structural elements is not sufficient to ensure security of a data base, for there are things which a user can do within his area of authorization that can generate sensitive information. These possibilities are discussed in the following two sections.

Data Aggregation

Individual data elements which are not by themselves sensitive may, in some cases, be aggregated from security sensitive information; e.g., data on individual combat units which is not classified and which is used extensively by personnel concerned with those units can be aggregated to form an order of battle. Statistical sampling is a well-known technique of providing discrete elements that are unclassified from a sensitive collection. It, too, is vulnerable to the aggregation of samples.

Adequate protection against improper data aggregation is difficult to achieve. Good techniques have not yet been worked out. They probably will involve identifying the data sets, which when aggregated become sensitive and limiting the aggregation that can be performed on them.

Another type of aggregation which can cause trouble by denying use of the data processing facility is called "data cancer". It involves inserting a program element that continually generates new data elements until the data processing system resources are saturated, with little or no resources available for legitimate users. A variant of this involves a program element that "puffs" itself up to fill system resources and after interfering with system use for a long enough time to cause trouble but not long enough to be correctly diagnosed, collapses and lies dormant for a while before beginning the process over again.

Inference

In some cases, certain functions of the aggregation of sensitive elements are not sensitive; e. g., the salaries of individual members of a group may be sensitive but the total salary of the group or the average salary may not. A problem arises when several of these non-sensitive functions can be combined to produce one or more sensitive elements; e. g., in the salary case, the total salary of a group of 10 people may not be sensitive and the total salary of a group of 11 people that includes the previous 10 may by non-sensitive, but the difference of these two is the salary of an individual, which is sensitive.

Other more elaborate statistical techniques may be employed to derive sensitive elements from unclassified aggregates.

ACCESS CONTROL

Security of a data base is dependent on control of the access of each user to the data elements and aggregates that he is authorized to use. It involves the coordination of several elements:

- user identification
- user profile - clearance, need to know, etc.
- input device - certain data may be accessed only from certain input devices; e. g., the list of password assignments may be accessed only from the security manager's terminal
- output device - certain data may be outputted only on certain output devices
- access privilege - read only, update, execute, append, private, unrestricted
- process - the process in which the data accessed is to be used; e. g., internal calculation only, direct output, remote transmission, etc.
- sensitivity - the sensitivity of the data accessed

Effective coordination of these seven elements can be accomplished in the following way. Each data element and structural element is assigned a sensitivity parameter value - e. g., a number or codeword - which is stored with it. Each value of the sensitivity parameter denotes an access rule which is a function of the six other elements. (Some access rules may depend on fewer than six; e. g., the access rule may be independent of the input and output devices). At each reference to a data element (or structural element) the value of its sensitivity parameter is checked and the access rule it denotes is applied. In some situations, access rules involving other elements may be needed; e. g., access may be limited by time-of-day.

The security sensitivity parameter may be stored with each data element, or the physical structure of the DMS can be designed so that all data elements having the same sensitivity are stored together in the same physical file. In that case, an implementation of the DMS will have at least as many physical files as there are security sensitivity parameter values. Then access control can be handled exclusively by the operating system, whereas in the first case at least a portion must be handled by the DMS.

APPENDIX IV

SECURITY VULNERABILITY AS A FUNCTION OF USER CONTROL OF SHARED RESOURCES

USER ISOLATION IN A SHARED RESOURCE ENVIRONMENT

It has been advanced as a working hypothesis that a "security perfect" computer system is vulnerable only to physical threats. In the real world of imperfect instruments, vulnerability is extended to include errors in design, implementation, operation and maintenance, and design and fabrication incompleteness to handle actual operational loads. It is then axiomatic that the more complex the operational system:

1. the greater the probability of error,
2. the greater the resources available to the interloper to probe the system for weakness, and
3. the increased sharing of resources increases the potential security exposure of the common user community to discovered system flaws.

Thus, if we can build better "firewalls" between users we can limit the extent of security compromise in multi-user, multi-level systems.

This appendix tries to increase our understanding of security failure modes and possible design strategies that offer promise of ammeliorating the security vulnerability of failure. The thesis advanced here is that better isolation of shared resources offers the best, and possibly only, solution.

We know from experience that software systems operating interpretively offer greater security than open-ended systems for just such reasons of restricted capability and isolation. As an example, this concept is the design base for the IBM operating system CP-67 that interpretively allows users access to "virtual machines" simultaneously sharing resources of a real computer. CP-67 is just one current model of such systems and it requires virtual memory hardware and software techniques not common in most operating systems. Alternative approaches are possible, even in current operating systems, by use of software interpreters. The tradeoff between the level of user capability and system vulnerability achievable is summarized in Figure IV-1.

VULNERABILITY INCREASE WITH INCREASED USER CONTROL

User control over the real hardware resources ranges from the user just watching a computer display, to total control of hardware where physical wiring can be modified. Figure IV-1 discusses these levels of control in terms of the resources shared, the direct vulnerability (1st level), and the security payoff to the interloper.

USER CONTROL	SHARED RESOURCE	1ST LEVEL VULNERABILITY	PAYOFF
1. Just Watch	Display Surface	• Malfunction/BUG/ Residue Access • Destruction/Jam • Sophisticated Jam	• Gain (Random) • Deny (Random) • Falsify (Random)
2. Initiate Program (1 + Limited Push Buttons) Manual Probes	• {OS, Appl. Prog.} CPU • Data STORE	• Insufficient Legality Check • Illegal Sequencing • Crash System	• Gain (Directed) • Gain (Random) • Deny
3. Transaction Only (2 + Enter Parameters) Machine-Aided Probes	• Time (Response Feedback) • Increased I/O Bandwidth	• Logic Path Complexity • Data Aggregation	• Gain • Deny • Falsify
4. Interpretive Code (3 + Code Sequences) Machine-Generated Probes	• Limited Psuedo-Machine (Interpreter)	• Higher Order Complexity • STORE Overload (Data Cancer) • CPU Overload (Program Loop)	• Gain, Deny, Falsify • Deny
5. Compiled Code	• Limited Real-Machine (Compiler)	• Break into Machine Code (see 6)	• Gain, Deny, Falsify
6. Machine Code	• Near-Total System Control • Real Addresses • Real Op Codes	• Violate Software (OS) Integrity • Incomplete System Design	• Gain, Deny, Falsify
7. Machine Code (Monitor State)	• Total System Control	• No System Checks & Balances • Modify Software (OS) Integrity	• Gain, Deny, Falsify
8. Hardware	• Total System Control	• Modify System Integrity	• Gain, Deny, Falsify

Figure IV-1. Increasing Security Vulnerability With Cumulative User Resource Control

User Control Levels

We conceive of these levels as consumer/resource transactions, i.e., a given level offers a set of resource capabilities that the user-consumer may invoke with commands, control language, or instruction dialogs. The closer the level gets to the physical machine, the more the dialog takes on the characteristics of a programming or machine language. Thus, we abstractly speak of the resource presented to the user at a given level as a "pseudo machine," and view the consumer/resource levels as a hierarchy of pseudo machines, each more closely approaching the real machine hardware. In actual implementation, a pseudo machine is a software interpreter that executes the user command instructions defined for that machine. Transaction systems for ticket or airline reservations are typical examples. Security is enhanced since the pseudo machine can be designed and implemented to perform any degree of checking and auditing of transaction requests (pseudo machine instructions).

Types of Shared Resources

Nearly all the real machine resources come into play indirectly even with the simplest user control options; however, this discussion focuses on the immediate resources the user can directly control, and which he shares with other users. These include I/O devices, Control Processing Units (CPU), and the system software -- Operating System (OS), Interpreters, Compilers, Data Management Systems (DMS), data files, and Applications Packages. We also consider "time" a shared resource for systems which feature responsiveness (scheduling, and real-time), feedback, and simultaneous access.

Threats and Vulnerabilities

For each level of user control, we examine the nature of the security vulnerability presented by the resources shared. For the very restricted user control levels, the vulnerabilities are "passive threats" resulting from accidental failure, or "active threats" on the physical environment such as destroying or debugging terminals. As the user's control over more resources increases, the active threats increase, enabling the user to probe for system weaknesses among the increasingly complex logic paths of the pseudo machine with which he is presented.

Security Exposure and Vulnerability Payoff

Unlike the commercial world where stealing CPU processing time, or use of proprietary products are major vulnerabilities, the military problem of security exposure to system vulnerabilities is essentially tied to the unauthorized "interference" with information communication to authorized users. This interference may take one of three general forms:

1) gain of information by an unauthorized user,

2) denial of information to an authorized user (e.g., crash computer system)

3) falsification of information to authorized users.

As the level of user control increases, the nature of the vulnerability payoff changes from random interference to more directed attacks to achieve specific objectives. This is not surprising as the user has more computer capability to assist in the probe.

DESCRIPTIONS OF USER CONTROL "PSEUDO MACHINES"

A "minimum model" of a secure computing system focuses attention on the causes of insecurity. The following is a minimum (useful) model:

1) Monoprogramming on a common set of hardware,

2) No cooperation (or presumed knowledge) between the programs, and

3) Limited shared resources; i.e., CPU, Main Memory, and OS.

This model is vulnerable to a number of threats:

1) Physical damage

2) Scavenging by passive browsing and residue pickup

3) Bugging the OS to do unauthorized spying or falsification.

Extending the model to include shared secondary storage (disc, drum, tape) and more sophisticated input/output capability (e.g., interactive teleprocessing) brings it to a level of practical usefulness and sets the stage for consideration of the eight levels of control summarized in Figure IV-1.

Just Watch

In this situation, the user can just watch a display of computer output, either a line printer or a display console. He cannot initiate any action; he cannot push a button. The only resource to which he has access is the display unit.

The most obvious vulnerability is that of an equipment malfunction or a software error that could cause information to be displayed to the user for which he is not authorized access.

Other vulnerability possibilities are that the user might, unnoticed, attach a "bug" or electronic sensing and transmitting device that would transmit the information being displayed at a later time. More sophisticated devices might also perform selective jamming of the display device, thus denying its use. Even more sophisticated but possible is a device which senses and transmits to an off-site computer the information being displayed. The computer processes this information in real time as it is received and sends false information to the device which causes that false information to be displayed.

A final vulnerability would be the possibility the user would leave a time bomb, which at some later time would destroy the display unit.

Initiate Programs

In this mode of user control, the user can initiate a program by pushing a button, but he cannot enter data. He has a limited number of programs to which he is allowed access. Since the programs which he can initiate may use any component of the computer, he has access to the entire computer system, except those programs to which he is denied access.

All the vulnerabilities of the "just-watch-mode" also apply to this mode.

New vulnerabilities arise from the possibility of inadequate access checking, which would then give him access to programs to which he is not authorized. Another vulnerability is that certain sequences of activation of the programs to which he is allowed access could leave certain computer registers in a state that would open a trap door to programs which he is not authorized access. Alternatively, some sequence could cause a system "crash", denying the system to other users.

Transaction Oriented

In this mode, the user is allowed to enter data as well as initiate a program; i.e., he may make "transactions." He may enter data, receive a response, and then based on that response may enter new data. With this "feedback" capability, he can engage in machine-aided sampling. In addition to feedback, the new resource that he has is the complexity of the data - the large number of data possibilities and the complexity of the data structure.

All the vulnerabilities of the "just-watch-mode" and the "initiate-program-mode" are present in this mode too.

A new vulnerability is that the complexity of the logic paths which the user can select or generate make it difficult to assure that one of them does not lead to a "trap door." A second vulnerability is that the aggregation of data to which the user is allowed access can lead to information to which he is not authorized.

Interpretive Code

At this level of control, the user can generate arbitrary code sequences for the psuedo-machine provided by the application interpreter, e.g., an interpreter for desk calculations or printing reports. This increased capability can be used to write code sequences that generate system probes to automate, and hence amplify the user's ability to try greater numbers and more complex thrusts at uncovering system weaknesses. In the least case he can deny authorized use by overload and saturation of shared memory and CPU services.

Compiled Code

The user at this level is closer to the real computer because he controls the machine via a sequence of real-machine instructions generated for him by the compiler. Legality checking of program "intent" is greatly reduced over interpretive controls of earlier control levels. Therefore, the user program may be able to break-out of high order languages (HOL) to directly attempt to exercise all hardware options as if user had direct machine code.

Machine Code (User State)

Most real machines restrict a portion of their machine instructions exclusively for the software monitor. User applications programs use the restricted instruction set and run under hardware control--User State--that traps illegal privilege instruction execution. However, the machine code user has nearly total system control because he can use and generate real op-codes and real hardware addresses. He can mount a major penetration attempt to violate the software monitor's integrity by exploiting incomplete system design--particularly the low level hardware/software interfaces.

Machine Code (Monitor State)

Monitor software is entirely defenseless against direct (deliberate or accidental) modification by itself or other monitor-state software, unlike the other levels which have both hardware and software support. The monitor software is most sensitive and complex portion of the system, so even minor modification can expose the system to major security compromise.

Hardware

As with level 7, this level permits users to modify the total system configuration of hardware and software.

APPENDIX V

PROCEDURE CONTROLS

INTRODUCTION

In contemporary computer systems, the effective use of procedures is critical to the implementation of any protection strategy. While significant advancements can be anticipated in providing more security control at the primitive level of system operations, the need for human controls and complementing internal procedures will persist.

For purposes of this discussion, a procedure is defined as an ordered means for controlling system operations. Associated with each procedure is a CUE (i.e., the stimulus which invokes the procedure), a DECISION (i.e., the process of solving a problem and determining an appropriate course of action), an ACTION (i.e., the exercise of control) and an EFFECT (i.e., the results of the action on the operation affected). This definition pertains to both the concept of procedure as human activity at the man-machine interface and the concept of procedure as a hardware/software convention.

PROBLEMS

Several problems can be identified with the use of procedures on current systems.

- The set of procedures employed is often not comprehensive, leaving pockets of uncontrolled system activity. A special case of this problem can occur when a system undergoes continuous change in terms of services to the user (e.g., adding new programming capabilities). If the control substructure remains constant, this succession of changes can uncover or create areas of vulnerability.
- Some procedures do not adequately control the object operation. Closely related is the problem of implementing the right solution for the wrong problem by being cued by some second-order effect of the real problem.
- Some procedures can be bypassed with impunity. Whether intentional or unintentional, people who effect procedures can often make leaps in executing a string of procedures, or they can ignore the procedure altogether.
- Procedures often have overlapping functions and jurisdictions, resulting in conflicts and confusion which neutralize or degrade the desired control. This is especially true at organizational interfaces where external procedures tend to be concentrated.
- The employment of procedures often creates problems of its own, thereby supplanting one problem (set) with another.

While specific cases of these problems tend to arise from subtle and complex sources, one can cite four generic source-factors which directly or indirectly contribute to most.

1. Procedures are adopted from a variety of authorities and for diverse and potentially inconsistent reasons. Some are required by formal policy or authority (often the same policy or authority established for manual information systems) and are implemented directly without being tallored for the particular context of the object system. Some procedures are adopted to fill gaps left by incomplete design and implementation of the system. Some are adopted to effect a management style which is intended to bring about smoother operations. Finally, some procedures are adopted to provide controls over controls. Each of these reasons may have validity for a given set of conditions, but unless it is implemented in the total context of the system, it can lead to many of the problems noted above.

2. Procedures are often omitted as a basic and integral design consideration. When they are considered in the design phase, they are often included as the "slack" variables and are consequently unstructured and underdeveloped. On the other extreme is the case of a formally required procedure which is often included as an immutable design element or constraint.

3. Procedures are often used to patch the system as an expedient measure. They are adopted under the "squeaky-wheel" principle, which says that a procedure is adopted only in response to a compelling problem and is implemented to solve only that problem. The cumulative effect of such implementations eventually imposes severe burdens on people and system overhead and can lead to subsequent implementations (or reimplementations) by least-cost methods which tend to be less effective.

4. Procedures are often built on inadequate technical foundations. Building a secure file access system into a data management system can be a futile exercise if the operating system allows a penetrator to gain control of the system.

PROCEDURAL AREAS

In confronting these problems, there is need to establish the system environments, by structure and function, in which procedures are applicable. The following (nonexhaustive) enumeration may serve as a point of departure.

- COMPUTER OPERATING PROCEDURES

    a. Console Operator Procedures

        (1) Start-up and shut-down

        (2) Normal operations (e.g. access controls, work flow regulation, etc.)

        (3) Recovery and restart

    (4) Accounting functions
    (5) Mode changes and conversions
  b. Software Maintenance
    (1) Update
    (2) Trouble shooting
  c. System Development
    (1) Testing and debugging
    (2) System integration
  d. Hardware Maintenance
    (1) Installation and changes of equipment
    (2) Trouble shooting
  e. Media Handling
    (1) Library media
    (2) Scratch media
    (3) Back-up media
    (4) Sanitization and degaussing
  f. Input Controls
  g. Output controls
    (1) Labeling
    (2) Dissemination
    (3) Destruction
  h. Operations Support

- PRODUCTION CONTROL
  a. Schedule of runs, remote access time
  b. Source media conversion
  c. Input validation
  d. Authentication of off-line user requests

- TERMINAL AREA CONTROLS
  a. Terminal area access control
  b. Terminal function/processing control
  c. Output controls

- CENTRAL FACILITY CONTROLS
  - a. Facility access control
  - b. Accounting for programs, run instructions, etc.
  - c. Housekeeping (e.g., trash disposal, etc.)
- PROGRAMMING CONTROLS
  - a. Requirements analysis and design
  - b. Coding
  - c. Testing and debugging
  - d. System integration
  - e. Documentation
  - f. Changes and patches
- USER AUTHORIZATION
  - a. User programming capabilities
  - b. System and data access limitations
- VALIDATION/CERTIFICATION
  - a. Hardware
  - b. Software
  - c. Auxillary physical resources
  - d. Procedures
  - e. Back-up resources
  - f. Total system
- SECURITY SURVEILLANCE
  - a. Instrumentation
  - b. Measurement
  - c. Contingency action
  - d. Reporting (including use of audit trails)
- CONTINUITY OF OPERATIONS
  - a. Redundant system configuration
  - b. Redundant application systems
  - c. Reconfiguration (including degraded operations)

d. Replacement resources (including back-up files, etc.)

e. Manual back-up

f. Conditions for reducing or suspending security controls

- VULNERABILITY ANALYSIS

  a. Establishment of acceptance criteria

  b. Analysis of environment

  c. Identification and assessment of threats and risks.

- SECURITY RESPONSIBILITIES AND AUTHORITIES

  a. Security officer

  b. Security support personnel (e.g., guard force)

  c. External liaison (e.g., local police, fire departments)

- MANAGEMENT AND ADMINISTRATION

  a. Planning controls (e.g., procurement cycle)

  b. Channels of communication

  c. Organization

  d. Quality control

  e. Personnel security program.

    (1) Selection of personnel (i.e., screening)

    (2) Indoctrination and training

    (3) Enforcement policy

    (4) Separation of duties and need-to-know controls

From this list of procedural areas, one can make a number of important observations. First, procedures pervade system operations and directly or indirectly affect nearly all system functions. Second, procedures exist in multidimensional layers, thereby posing complex interfaces to each other and among the agents which exercise them. Third, all entries in the list may be implemented by personnel procedures, hardware/software procedures or a combination of the two. Very few entries, however, imply a method of implementation by the mere recognition of their necessity. Implementation is a function of a system context.

RECOMMENDATIONS

Clearly, much work is needed to enhance the use of procedures in securing any system. It is inconceivable that any development project can contribute to the

solution of problems associated with any form of security, let alone multilevel security, without incorporating procedural elements. Considerations discussed above should pervade all such efforts.

These are, however, three areas of development work associated with procedures which should be promoted separately.

1. Formal studies of procedures are needed to determine how procedures constitute vulnerabilities as well as countermeasures. The following questions are exemplary of the concerns which should be addressed: In exercising a procedure, can (and with what likelihood) an individual receive spillage of data for which he has no need-to-know? Is his span of control derived from a procedure greater than his assigned functions? What is the likelihood that an error on the part of an individual exercising a procedure will result in a security incident? It is recommended that a procedural analysis of a major, supposedly secure system be undertaken to refine these questions, derive others and build an empirical procedural model for independent study and evaluation. Once this is accomplished, a general theoretical model should be developed to support behavioral studies in human engineering, facilities management, etc. as well as technical studies in developing hardware/software security measures.

2. Work is needed to develop a stronger technical foundation for procedures. Much of this can be accomplished if the developer identifies at an early design phase the areas of the system where procedures are needed. What is there about the intrinsic data flows and processing operations of a system that promotes or inhibits security? What security measures are intrinsic to the system and what must be added? In terms of adjunctive security measures, work is needed to determine which internal procedures can be externalized and, conversely, which external procedures can be internalized. The purpose here is to find the domain over which maximum effective control can be exerted while maintaining system integrity, economy and user/operator convenience. An important part of this study should be to determine the relevance of procedures derived from manual information processing systems for ADP systems, and additionally, which procedures are required in ADP systems that transcend those of the manual systems. Another consideration should be how to make security procedures natural to the user (for external procedures) and to the system (for internal procedures), and where such procedures should be transparent. Related to this consideration is the question of how best to control the access to and use of procedures. In all of these considerations, it is necessary to distinguish between gadgetry and good practice.

3. Work is needed to establish pertinent procedural requirements and guidelines. As a starting point, this work should set forth security standards in programming and documentation procedures, console operator procedures, I/O controls (including generation and distribution), and management security

guidelines. These requirements and guidelines should include principles and instances of acceptable practices. The following are indicative of such principles.

A. Each procedure should provide for individual accountability by identifying the individual exercising the procedure with the action and effects of the procedure, by authenticating an individual's authority to invoke a given procedure, and by providing for evidence of the fact that a procedure was effected (e.g., audit trail of procedures).

B. Each procedure should be established within appropriate need-to-know limitations. This means that the range of procedural actions and effects is a function of a person's job and nothing more. It also means that each procedural action and effect should have official sanctions.

C. Procedures should have functional integrity. This means that unstructured procedures (i.e., procedures invoked by ambiguous cues or by independent initiative, effected by arbitrary decisions and actions, and producing undefined and unpredictable effects) should be minimized. Where such procedures are necessary, however, some variation of a two-key system should be employed. Functional integrity also means that failsafe/failsoft mechanisms should be incorporated into procedures. Finally, it means that measures are implemented in such a manner that an individual will employ only those procedures necessary for a particular level or type of security mode.

APPENDIX VI

IMPACT OF TECHNOLOGY ON SECURE COMPUTING SYSTEMS

The prognostication business is at best a risky one. We find that often our prophecies are either too conservative or too outlandishly optimistic. More often than not they are both at the same time. For example, who could have foreseen the tremendous growth in the computer technology 15 years ago? The impact of the integrated circuit on computer size, cost, and speed has been staggering. At the same time, however, computers have yet to live up to the promise held forth in the 50's and 60's. Much of this is due to the problems of system design and software implementation. In this section, some of the more promising technological trends are indicated, along with the probable report of the resulting technology on the problem of computer security.

Perhaps the most interesting potential of modern technology will be the radical reduction of cost of computer main frames. We can, for all intents and purposes, assume the computer main frames will be effectively "free" in the not too distant future. As a consequence, if it is really necessary to separate various users, each can be given his own computer. However, more often than not they are dealing with common data bases and must hand off certain common data to one another and, on occasion share programs. As a result, we are still in need of secure computing systems. With very low cost computer logic however, we have the possibility of a distributed system. By this is meant a system in which the various system functions may be distributed among different machines which are "netted" together. Netting does not imply a number of machines doing identical tasks, nor does it imply a number of necessarily identical machines. Each machine has its own unique task. Examples of these might be (1) file system machine; (2) a communications processor; (3) a set of user machines each performing tasks for their own specific users. Distributed system could take many forms. Questions remaining include: what is the best system for interconnecting them; what are the unique security problems posed by such a "localized" network, etc.

Another result of the very low cost of future logic will be the ability to include rather sophisticated pieces of logic as parts of any CPU. Thus techniques for secondary file encryption, encryption of data in primary files, etc., are all feasible and possible. In fact, it might even be feasible to enforce certain types of data handling discipline that could be implemented by means of an appropriate chip or chips that by statute must be a part of any public system. Whether this last would be a practical solution or not remains one of the subjects of applied research in this area.

Up to now, large scale systems have been designed by ad hoc methods. Where automated design aids existed, these tended to impose some structure on the design, but there was no guarantee that the structure would be followed in implementation. Computer-based design technology, found both here and abroad, has been directed to formalizing both the design and implementation of large scale systems. The results

of these efforts indicate an order of magnitude increase in the integrity of the designs. This improvement comes about because of the ability, using a computer, to test and validate the design before implementation, and the elimination of human interaction during critical implementation steps. The projected effect of this technology on security is large since it will be possible to produce validated hardware and software designs before implementation, and even permit economical Government development of its own designs for special applications.

A final trend worth noting is the growing interest in Declaration on Goal-Oriented Programming. This trend (also known as Implict Programming, Automatic Programming, or Heuristic Programming) is an important concept. Under this concept, the programmer no longer defines the method (how) for performing a function (or program); rather he specifies what must be done. The methods by which this kind of programming will be achieved appear to be based on recognizing the stereotypic nature of much programming, and applying program generation techniques to provide the tailoring of the application to the functional requirements. If the method (generators or whatever) is certified, it could eliminate the malicious user threat from most systems. It appears that these concepts will take ten years to become assimilated technology, although some initial results will be available earlier.

APPENDIX VII

AIR FORCE COMPUTER SECURITY TRENDS AND PROBLEMS

INTRODUCTION

This appendix reports upon the trends and problems in computer security which were identified by the Requirements Working Group supporting the Computer Security Technology Planning Study Panel. The information included was gathered by the Working Group through briefings and discussions with representatives of the individual commands and through reference to documents provided by the commands.

The composition of the working group varied among the different instances, but generally consisted for several members of the study panel and one or more working-level staff officers representing the subject command. The appendix consists of a series of sections each describing the information gathered about the computer system security needs of one of the following organizations:

Air Force Logistics Command (AFLC)

Air Force Data Services Center (AFDSC)

Satellite Control Facility, Space and Missile Systems Organization (SAMSO)

North American Aerospace Defense Command (NORAD)

Air Force Communications Service (AFCS)

Air Force Global Weather Center (AFGWC)

Strategic Air Command (SAC)

Military Airlift Command (MAC)

Electromagnetic Compatibility Analysis Center (ECAC)

Tactical Information Processing and Interpretation System (TIPI)

Air Force Security Services (AFSS)

(Although AFSS participated in the working group discussions, insufficient applicable data were gathered for inclusion in this appendix).

In each section an overview is given identifying the command, its major responsibilities and the command's representative(s) on the Requirements Working Group. Then follows a brief description of the system pertinent to the command's computer security requirements, and a discussion of their current information security problems. Next, the current perception of future trends in the command's security problems and capabilities is given, followed by their present and planned solutions to computer security problems. The material in this appendix was used as a basis for the statements made in Section II of this document.

AIR FORCE LOGISTICS COMMAND (AFLC)

OVERVIEW

The Air Force Logistics Command (AFLC) purchases, manages, and distributes material for the entire Air Force. This function is a 10-15 billion dollar a year operation. The AFLC representatives on the Requirements Working Group were Capt. Ted Legasey and Mr. Walter Schull (ACTA). Both are involved in the design and implementation of the command's Advanced Logistics System (ALS).

SYSTEMS

AFLC is now designing and implementing the Advanced Logistics System (ALS). The motivation for this system is to achieve increased economy and responsiveness through the establishment of a uniform logistics data base and uniform, updated computer facilities. ALS will provide complete inventory control and distribution management throughout AFLC. ALS will include a computer center at each of AFLC's six Air Material Areas (AMAs) plus one additional center to support the Nuclear Ordnance Logistics System (NOLS). The NOLS Center is isolated because of the sensitive data it processes, but the other ALS Computer Centers will be interconnected by AUTODIN. Each center uses a CDC Cyber 70 multiprocessor computer with 1 million characters of main memory and 22 billion characters of immediate access (secondary) storage. Each AMA will have 400 to 500 remote terminals and will include nine different types of equipment:

1) High speed card readers.
2) Low speed card readers.
3) High speed card punches.
4) Low speed card punches.
5) Keyboard printers.
6) Cathode ray tubes.
7) Data Collection terminals.
8) Receive only typewriters.
9) High speed character printers.

The ALS basic software packages are being developed both by contractors and AFLC personnel. The contractor software packages are the Executive/Monitor System (EMS), which is the operating system, and a Data Management System (DMS), to handle the Unified Data Base. AFLC has written the basic specifications for both the EMS and DMS. Neither the EMS nor the DMS will be an off-the-shelf package. The AFLC-developed software packages include the Central Control System (CCS) and the test systems used to test the applications programs. The CCS forms the interface

between the applications programs and the contractor developed DMS and EMS. CCS has four basic functions: Input, Output, Management and Control, and System Control Support. One part of the System Control Support function is security.

The ALS software implements a transaction-oriented system designed to support the AFLC item managers. Its application programs are written by AFLC programmers. The application programs use structured programming with a hierarchical structure of processing in which each of the 35 current major subgroups is termed a "logistics process." Each logistics process is made up of "events," each of which is the piece of processing logic required to carry a particular transaction from beginning to end. At present there are 1086 such "events" in the system. Each event is made up of one or more "modules," each of which has a maximum length of 10K words. Presently the system contains 8,683 modules which are written in COBOL. The estimated maximum workload at a site is 27 transactions per second.

SECURITY

The ALS with the exception of NOLS, handles a small amount of classified information (less than 1% of the ALS data base is classified; 90% of this is classified Secret, and the rest is Confidential). The NOLS is isolated from the balance of ALS and is required to handle nuclear weapons information up to Top Secret Restricted Data. The isolation of NOLS from the remainder of ALS is necessitated by the fact that the security of the ALS computer system is not deemed adequate for protecting the highly sensitive data in NOLS.

Each ALS central computer facility must be capable of processing classified information in either a random or batch-sequential mode. Job scheduling will be used to permit concurrent processing of both classified and unclassified applications. Classified information will be transmitted between a Central Computer facility and its associated remote devices only when the circuits are either approved or encrypted. Both secure and nonsecure remote devices will be allowed to operate concurrently during processing and transmission of classified information.

While the terminals and item managers that handle classified data are cleared, the bulk of the ALS terminals are not cleared. The restriction of access to classified data is the responsibility of the ALS computer hardware and software. User access to classified information will be controlled through the use of passwords. All ALS computers are housed in secure environments. The programmers who develop the ALS applications programs and the CCS hold Secret clearances. Those working in NOLS hold Top Secret clearances.

FUTURE TRENDS

Although no formal planning activities exist in some of the following areas, they will potentially impact the security requirements. It is expected that the ALS will be expanded in stages both to increase the scope of services it will provide and to increase the interconnection with other systems. It is possible that the ALS will be

tied into the base level supply and support systems and further that it will be more closely tied into the logistics systems of the other services. The amount of resources that the ALS will handle — greater than 10 billion dollars a year — is enough that even a small portion of them would constitute a worthwhile target. Protection against the unauthorized appropriation of resouces may require an eventual increase in the scope of the security system; i.e., to include more than just protection of national security, (classified) information.

SECURITY SOLUTIONS

The main source of software security controls in ALS is the central control system (CCS) software. Its security related functions are generally divided into two classes: the "preprocessing function" and the "threat monitoring" function. The preprocessing function includes the establishment and maintenance of security related tables and the use of these tables and other information to check and validate any programs prior to their entry into the operational program library. The threat monitor function consists of: a) identifying and authenticating each user and program request to access and process classified information; b) communicating security classification information to the EMS and DMS; c) insuring that classified output is transmitted only to secure output devides; and d) monitoring and audit trail and requesting job termination of programs when abnormal conditions arise during classified processing.

AIR FORCE SATELLITE CONTROL FACILITY (AFSCF)
SPACE AND MISSILE SYSTEMS ORGANIZATION (SAMSO)

OVERVIEW

The Air Force Satellite Control Facility (AFSCF), of the Air Force Systems Command's Space and Missile System Organization (SAMSO), is the DoD Agency responsible for the management, design, operation and maintenance of a worldwide information network for the monitoring, testing, control, and support of space satellite operations. The network is comprised of ten remote tracking stations (RTS) located at seven geographically dispersed sites and a Satellite Test Center (STC) located at Sunnyvale, California. Within the STC are several Mission Control Centers (MCC) which are individually assigned to operationally support the space mission programs by: (a) tracking the satellite from the appropriate RTS; (b) maintaining currently updated satellite ephemerides; (c) monitoring vehicle health through the reduction of telemetry data; and (d) commanding the satellite to perform specific functions. The AFSCF representatives on the Requirements Working Group were Major John Marciniak (AFSCF/DMD) and Mr. Tom Carr of Aerospace Corporation.

SYSTEM

The main elements of the AFSCF network are the RTSs and the STC. Each RTS contains the transmitting, receiving, and tracking equipment necessary for the reception of telemetry data, satellite position determination, and the commad of satellites as they pass through the RTS coverage. Individual teams operate the RTS antennas, the telemetry equipment, and the data systems. These teams report operationally to the Operations controller, who is in voice communications with the appropriate Test Controller, located in the MCC assigned to the mission.

Telemetry tracking and commanding data pass between the STC and RTSs through computers at each facility which control the data interchanges via 2400 bit-per-sec (bps) and full duplex communications lines. These computers consist of a Univac 1230 at each RTS and CDC 160As at the STC.

The program office and support personnel at the STC plan, integrate, schedule, and control the activities of the total AFSCF network for the support of a multiplicity of satellites and satellite programs. Each satellite program office has its support team that operates from an assigned area at the STC, including a specifically assigned MCC. That MCC is kept constantly in communication with the appropriate RTS(s) through a communication switching system and one of the CDC 160A "Bird Buffer" computers. These latter computers are assigned and operated on a schedule that is pre-planned to meet mission requirements by a Network Control Group. Except in cases of scheduling conflicts that must be coordinated with other programs, each MCC and support team can operate as though it were serving the only program using the AFSCF network.

The program support teams are composed of: (a) Support personnel in direct communication with the RTS(s); (b) The Field Test Force Directors (FTFD) and program directors who provide operational guidance; and (c) a staff of program-oriented specialists who constantly review satellite status and plan future activities.

The computer complex at the STC includes five CDC 3800 computers that are operated off-line by the program support teams to assist them in such activities as mission planning, command operation, and ephemeris determination.

The system's interaction with the satellite occurs in three phases: Prepass, Pass, and Post-pass. These terms refer to the satellite's position with respect to the RTS's coverage pattern.

When a satellite pass is anticipated the satellite is assigned to an RTS and the Pre-pass phase is begun. The Pre-pass activity consists of preloading the RTS with satellite location data and commands and queries for the satellite. The commands and queries have been generated off-line on a CDC 3800 Flight Support Computer. The Pre-pass data is transmitted by the assigned 160A via the 2400 bps line to the Univac 1230 at the RTS.

During the pass, which may last from minutes to hours, queries and commands are sent from the RTS to the satellite, and telemetry and tracking data are received from the satellite. The data received from the satellite is preprocessed on the RTS's 1230 and transmitted to the 160A at the STC. The STC has a display facility which contains 150 lpm printers and graphic plotters. The printer output is sent to several locations around the facility by closed-circuit TV. In addition the 160A writes a Bird Buffer Recording Tape (BBRT), recording all the telemetry and antenna data from the pass. After the pass is completed, the remaining data from the pass is transferred from the RTS 1230 to the 160A and the BBRT is completed. After its completion the BBRT is moved from the 160A to one of the 3800s for further (Post-pass) data reduction and the generation of commands and queries to be used on the next pass. These commands and queries are then entered on another tape for transfer to the proper RTS during the next Pre-pass phase.

The ephemerides of satellites and spacecraft generally feature a precession of the ground tracks of their subsequent passes around the globe. Thus a particular satellite will usually pass through the coverage of several different RTSs at different times in its orbital history. To accommodate this and other factors there is a switch that allows connecting any 160A to any particular RTS 1230 for a particular satellite pass. Each of the MCCs connect to the assigned 160A Bird Buffer by one switch and then to the appropriate RTS 1230 by another switch.

SECURITY

At the STC most of the data on the 160As is unclassified. However, the data passed to the RTSs for Pre-pass loading of data and commands is sometimes classified up to Secret. Most of the real-time data that is exchanged during a pass is minimally

classified and most of the data in the MCCs is unclassified. The problems associated with data exchanges between the MCC and the RTS and satellite appear to center more on protection against misrouting of data than against security compromise. Most of the data handled on the 3800s is sensitive and classified but it is protected by the fact that each of the 3800s operates off-line in a separate lockable room.

FUTURE TRENDS

Plans are under way to upgrade the SCF. Two plans have been considered. In the initial upgrading plan a triplex of IBM 360/67s was to go into the STF in 1965. The 360/67s would do on-line work, interacting with the 1230s at the RTSs, loading problems into any of five CDC 3800s and providing control and display capabilities to the MCCs. This plan was withdrawn when the Manned Orbiting Laboratory (MOL) program was terminated.

The current plan is to replace the 160As by an equal number of microprogrammed machines and emulate. A microprogrammed circuit switch will tie the new machines to peripherals located, as now, in the MCCs. The 3800s will continue to operate off line with each in a separate lockable room.

SECURITY SOLUTIONS

Little information was given concerning specific security solutions, which exist or are planned for the AFSCF. Most classified information handling is done off-line on the CDC 3800s in secure areas. The other concentration appears to occur in the commands from the RTS to the satellite which are generated during the Pre-pass and often contain sensitive classified information. SAMSO gave no information concerning security solutions for that information.

AIR FORCE DATA SERVICES CENTER

OVERVIEW

The Air Force Data Services Center (AFDSC) operates a major multi-computer service bureau located in the Pentagon. The Center provides data processing support to Headquarters USAF and the Office of the Secretary of Defense within the Pentagon and the Washington D.C. environs. The work includes such diverse areas as payroll processing, responses to congressional actions, invocation of DOD budget information, and the running of extensive models and simulations. There were 380 people on the Center's staff in the Spring of 1972 and the number was expected to increase. Captain Wah Leong was the AFDSC representative on the Requirements Working Group.

SYSTEMS

AFDSC had in the Spring of 1972 two Honeywell G-635 dual processors, each assigned 180 million words of online disk storage and 24 tape drives. They have 9 secure remote vaults that contain secure terminals and crypto gear. Each vault contains at least one G-115 remote batch terminal and two Terminet 300 teletype terminals for time-sharing use. In all the 9 vaults contain 22 secure terminals. The 635 systems have been generating about 4 million printed pages a month. AFDSC also has an IBM 7094 which is dedicated to the processing of Top Secret information.

In addition to the classified system AFDSC has been spending $40K to $50K per month for the use of commercial, unclassified, time-sharing services, including GE Mark II and others. At present there are 27 terminals using these commercial services, but AFDSC has need to use more terminals.

The Data Services Center users submit their programs in COBOL, FORTRAN, and, to a lesser extent, in assembler language for local batch processing. Remote batch jobs may be entered via any of several high speed terminals. The GCOS III time-sharing system (TSS) provides the users with two kinds of capabilities:

(a) remote interactive programming in FORTRAN and BASIC; and

(b) a low speed path for the entry of jobs.

The uses of the system span a wide range of programming languages, size, and complexity. While some of the users require production runs of periodic reports, there are many new programs constantly under development by various users.

SECURITY

The Data Services Center handles data ranging from Unclassified through Top Secret. The data handling is separated into three categories, each with a separate method to protect its security.

A dedicated 7094, operated in a closed environment, handles all Top Secret information processing.

The other levels of classification, unclassified through Secret, are handled on the G-635s, which are located in a closed Secret environment. All terminals, including any remote terminals, which connect with this closed environment are required to be secured up through Secret, and all users have Secret security clearances. All products of the system are given a tentative classification of Secret until they have been given a permanent classification by the user. These criteria apply to all accesses to, and products from, the G-635 systems, including those associated with remote processing of unclassified information.

The third category applies to certain users who require solely unclassified time-sharing services for limited computational tasks. These users have nonsecured terminals and uncleared operators that cannot be permitted access to the closed Secret environment. For these users the Center purchases G-635 time from commercial service bureaus, at a cost which is increasing from $200K in 1970 to an estimated $500K in 1975.

A further economic problem associated with security is the cost of the remote secure terminals. Currently, each of the secure remote sites costs $50K to build; there are nine in existence and several more are planned.

FUTURE TRENDS

AFDSC eventually plans to convert the G-635s to provide multilevel security in order to eliminate the use of commercial time-sharing service and its attendant costs. As a first step, they will probably set up a separate in-house system which will be dedicated to unclassified use. This system will be served by a separate Datanet 30 providing dial-up capability for the unclassified lines. It would be segregated from the other systems. With this separate unclassified system the multilevel security problem will be avoided. A major problem that will remain for the unclassified systems users is privacy, since the system will be used by unsecure terminals operating through open communication lines.

It is expected that the entire Data Services Center computer area will eventually operate with multilevel security up to Top Secret.

SECURITY SOLUTIONS

At present, there is no software solution available for open multilevel secure operation of AFDSC's computers. Consequently the center operates with the three-category approach described under "SYSTEMS", above: (a) a dedicated IBM 7094 for Top Secret; (b) the G-635's operated as a Secret system for Secret through unclassified; and (3) commercial time-sharing contracts for completely unclassified work. The AFDSC computers are dedicated to operation for cleared users, while uncleared users must be served elsewhere. In some cases a secure remote facility has been

implemented using secured communications to allow cleared users to do unclassified processing on the Center's computers. This is a very costly, and seemingly untenable approach, costing $50K initially for each secure terminal (currently a total of $450K) plus the continuing costs of servicing the crypto gear and the other security requirements of the terminals. This approach has been dictated by the inability of the GCOS III operating system to resist penetration attempts by uncleared programmers.

NORTH AMERICAN AEROSPACE DEFENSE COMMAND (NORAD)

OVERVIEW

The North American Aerospace Defense Command (NORAD) has headquarters located within the NORAD Cheyenne Mountain Complex (NCMC) in Colorado. NORAD is a multi-national command including forces of both the United States and Canada and is responsible for the aerospace defense of the North American continent. The electronic command and control system for the Commander-in-Chief of NORAD (CINCNORAD) is located in the NCMC and aids him in fulfilling his command responsibilities. CINCNORAD has an additional role as Commander of the United States forces' portion of NORAD, termed the Continental Aerospace Defense Command (CONAD). Sensors located all over the world gather information about aircraft movements, missile launchings, man-made objects in space, weather, status of forces, and intelligence data. The information flows in via a multiplicity of communication channels to computers that evaluate the data, sifting the significant from the trivial, and present the significant data as quickly as possible to aid the commander in his decision making. The representative of NORAD on the Requirements Working Group was Captain Paul D. Carr.

SYSTEM

The NORAD system discussed was the data processing and display portion of the electronic command and control system used by CINCNORAD. This system receives information over multiple routes from the Ballistic Missile Early Warning System (BMEWS), the Distant Early Warning (DEW) line, the Space Detection and Tracking System (SPADATS), the Air Force's Weather Observing and Forecasting System, overseas warning systems, intelligence gathering systems, and the Bomb Alarm System. This input information is processed for threat evaluation and presented to the Commander of NORAD, the Canadian Forces, the National Military Command Center, and Strategic Air Command. The NORAD command post is built around a highly sophisticated wall size display system. The Commander and his staff occupy three levels of the command post looking out over the 12 x 16 foot screens. In addition, 15 individual CRT display consoles serve the personnel of the command post. The system is now being up-dated and expanded. Two World Wide Military Command and Control System (WWMCCS) computers are presently being installed in the NCMC. One of these will serve as the Space Defense Center Computer (SCC) and the other as the NORAD Computer System (NCS). The latter machine will also do all of the utility work for NORAD, all of the Aerospace Defense Command's (ADC) logistics support processing, and ADC's force reconstitution processing during the post battle phase. There are to be two remote terminals both of which are to be located in secure areas.

SECURITY

At the present time, the NCMC processing is classified Secret, but with the addition of another machine, some excess time may be available to devote to war-gaming

and other Top Secret activities. Only two remote terminals are planned and they will be located in secure areas. Some special access material is used so the requirement for protection of Special Access Required (SAR) information is present. Intelligence information must be handled separately from the general NORAD system, by the WWMCCS computer for the Intelligence Data Handling System (IDHS) of CONAD. There is a possibility of a cross-tie between the CONAD IDHS and NORAD systems in the future which would increase the classification of the NCS to Top Secret. Both systems are closed with all terminals located in secure areas. However, some of the incoming communications lines are not secure.

FUTURE TRENDS

The connection of the NORAD system to the intelligence network may bring an increase in the complexity of the security problems. The fact that the response time of the NORAD system is critical would cause an impact on any security system that carried much overhead with it.

There will be new CRTs with the WWMCCS machines. The CRTs are now driven by the Display Information Processor (DIP). This processor was purchased as part of the BMEWS and is a very high reliability machine. It is not clear whether the DIP function will be replaced by a WWMCCS machine.

There are plans to tie the SCC, NCS, and IDHS machines together. WWMCCS GCOS and the WWMCCS security package will be used. The WWMCCS security package will be checked by JTSA and DIA. However, NORAD will still have to evaluate it against their security and performance requirements.

SECURITY SOLUTIONS

Little was brought out concerning security solutions. The NORAD system is closed with all elements located in controlled areas. The software security package will be that of the WWMCCS and not much information was known about it at the time of the working group meeting. The security package will have to be evaluated against the detailed requirements that NORAD did not point out at the working group meeting.

AF COMMUNICATIONS SERVICE (AFCS)

OVERVIEW

The Air Force Communications Service (AFCS) is charged with the responsibility of insuring that responsive communications systems are developed and operated for the Air Force. The AFCS representatives to the requirements working group were Captain Raymond D. Suffron who presented information on the Local Digital Message Exchange (LDMX), and Captain Bob Flechtner, Lt. N.L. Mejstrik, and Mr. Gene Snell who presented information on the automatic Digital Network (AUTODIN). The LMDX is a planned system while the AUTODIN is a system that has been in actual use for several years.

SYSTEM

AUTODIN is a general purpose store-and-foward communication system that consists of switching centers, terminals, and communication lines. The system handles about ninety to one hundred thousand messages per day per switching center with the messages averaging 32 eight-character line blocks in length.

There are eight switching centers in CONUS and several overseas. There are two kinds of switching capabilities in the switching centers: message switching and circuit switching. Only three of the centers have circuit switching capabilities, and only one of the three is now active, AFLC being the user. The other switching centers have only message switching capability, with messages processed through the switching center on a first-in-first-out basis by priority. At the present time, RCA machines are used in the CONUS AUTODIN switching centers, while Philco-Ford machines are used for the overseas switching centers.

Switching center software is controlled by the Defense Communications Agency (DCA). The programs are written in machine language and DCA sends changes to the programs out to the AUTODIN sites. Changes to the AUTODIN programs are made by and checked by teams of two programmers. The old and new program tapes are compared by another program to see what alterations were really made. Online test are also made. Four people check the changes: two programmers and two operators.

SECURITY

AUTODIN, being a store and forward system, is rather simple and straightforward from the standpoint of security.

Protection of message security depends upon the physical security of the dedicated machines at the switching centers, encryption of the dedicated external communication trunks, and software verification by the switching center that each message is routed only to properly cleared destinations.

The header of each message in AUTODIN contains routing indicators and an indication of the classification of the message. The switching center software performs such functions as checking tables stored in the system to see whether each of the receivers indicated by the routing indicators is cleared to receive information of the designated classification. If the receiver is cleared, the message is delivered; if not a service message is sent back to the originator indicating this fact. Basically, the process of checking the routing indicators against the classification of the receiving terminal does the job of protecting the security of the message. The executive systems used are specifically tailored for AUTODIN by cleared contractors and in-house programmers.

The AUTODIN system is cleared up to Top Secret but it is connected to a variety of terminals including unclassified as well as secured terminals.

FUTURE TRENDS

At the present time AFCS is evaluating the ARPA sponsored network of computers (ARPANET). Their objective is to determine how effective and efficient the ARPANET can be in terms how much traffic can be passed through it. The ARPANET is not at present secure so only simulated tests will be run.

A system that is planned for the future is the Local Digital Message Exchange (LDMX). This system is intended to consolidate transmission and distribution of all air base digital communications in a single exchange. Current practice is to have several independent communication centers on a base: the base communication center, the command and control center, and possibly others. Connections to AUTODIN or other external communication systems are handled independently for each of the communications centers. The external connectivity pattern followed varies from base to base and often results in inefficiencies such as the duplication of facilities. The first goal of LDMX is consolidation of these several facilities into a single communications center per base.

LDMX will be an on-base store and forward switch, one at each of eighty locations around the world. Each LDMX switch will serve a multiplicity of terminals on the base and perform the necessary store and forward interfacing of messages between the terminals and the AUTODIN access circuits. Each LDMX switch will be connected to two separate AUTODIN switching centers. In addition, LDMX will provide for interconnection and interfacing among the terminals on the base.

When implemented LDMX will operate about as follows: each message will be entered, via one of the remote terminals at the base, into the LDMX store and forward switch; there it will be automatically routed, its format changed into JANAP 128 format, and it will be forwarded to the AUTODIN switch. The AUTODIN switch will then forward it to its final destination. If the destination is equipped with an LDMX switch, the message will be routed directly to the addressed terminal. The LDMX switch thus will do automatically the work of on-base message routing currently done by the communications center personnel. The LDMX switch will be able to handle

normal message traffic as well as command and control traffic and computer to computer transmissions for systems such as MAC Integrated Management System (MACIMS). Thus the LDMX could be the only communications processor on each base.

The LDMX switch could accommodate a large number of secure terminals on each base. A problem that is being addressed currently is the development of both low and high speed secure terminals for use in LDMX. The cryptographic and protected areas needed to accommodate these terminals are currently quite expensive. Therefore a program involving AFCS, ESD, and NSA is now in existence to develop secure terminals for use in a store and forward system. The cost goal is $5,000 each including the cryptographic equipment. Present plans are to obtain about 100 terminals at each of 30 bases for a total of 3,000 terminals; about 2400 of these would be low speed and 600 would be high speed terminals. The projected time for system installation is about 1980.

SECURITY SOLUTIONS

The AUTODIN system relies on checking the classification authorized for each circuit. The authorized classification is stored in tables which are prepared off-line and then entered into the computer. Changes to AUTODIN software are prepared at a central location, then checked indepdently several times before being put into use.

The LDMX system is developing a low cost, secure terminal for use in a store and forward environment which will have the cryptographic equipment and terminal hardware as an integral unit. The cryptographic equipment will be protected by a secure enclosure that is part of the terminal. Both high and low speed terminals are being developed.

AIR FORCE GLOBAL WEATHER CENTER

OVERVIEW

The Air Force Global Weather Center (AFGWC), located at Offutt AFB, Nebraska, is a named USAF unit under the command jurisdiction of the Commander, Air Weather Service. The AFGWC provides meterological support to military command and control systems, including the National Military Command System (NMCS) and the Strategic Air Command Automated Command Control System (SACCS), Air Weather Service units and special mission aircraft as directed by the Air Weather Service or higher headquarters. In addition, special environment and support is provided to classified projects for USAF and other US government agencies. The AFGWC representative on the requirements working group was Lt. Col. Charles R. Stevens.

SYSTEM

The major automatic data processing equipment of AFGWC consists of four UNIVAC 1108 multiprogrammed computers integrated into a single system. The four computers will be interconnected by Inter-Computer Coupler Units (ICCU) so that there can be two-way exchange of data between systems I, II, and IV, and one way receive-only input into system III from system I. Conventional data are gathered and routine products are disseminated by the AWS-AFCS Automated Weather Network (AWN) which interfaces with the AFGWX system. Reliability is achieved by equipment redundancy. Each of the four UNIVAC 1108 computers may operate in real-time, providing information either upon request or at regular intervals.

The functions of each of these computer systems are discussed below:

a. System I. This system is the applications processor. Its main functions are: communications, decoding and validation of observed meterological data, dissemination of tailored products, generation of computer flight plans, and input to Strategic Air Command Automated Command Control System (SACCS). When System I is down, system IV becomes the logical System I.

b. System II. This system is the meterological processor and provides backup to system III when needed. The analysis and forecasting models are executed on this system for the purpose of building the data base for use by System I.

c. System III. This is the Special Projects processor which provides support to classified projects and processes meterological satellite data. This system is currently being approved to process Special Access Program material.

d. System IV. This system is the backup and development processor. Its main functions are to provide backup for system I or system II and to test developmental software. In addition, system IV processes some scheduled production which cannot be accomplished on system I.

The principal components of the system program package for AFGWC are the UNIVAC EXEC VIII and the Real Time Operating System (RTOS). EXEC VIII is a generalized control program of about 40K 36-bit words developed and maintained by UNIVAC. It provides multiprogramming capability, three modes of operation, peripheral equipment control, and numerous other services of value to the user.

RTOS is a supplemental program developed by AFGWC and maintained by AFGWC and AFCS programmers. RTOS provides the interface between the specific technology of the environmental data processing programs and the generalized services of EXEC VIII. In addition, RTOS initiates scheduled batch mode programs, identifies and manages incoming data, accepts and services requests for initiation of demand programs, and routes products to the appropriate output devices. AFCS programmers will eventually assume responsibility for the maintenance of the communications modules of RTOS. They also develop software for the forthcoming direct interface of the AFGWC computers with AUTODIN.

The UNIVAC 1108 communications subsystem enables the 1108 to receive and transmit data via any common carrier at any of the standard transmission rates up to 40.8 K bits per second. The subsystem consists of two principal elements: the Communication Terminal Module (CTM) and the Communications Terminal Module Controller (CTMC) through which the CTM accommodates two full duplex, or two input simplex communication lines. The CTMC can handle 16 CTMs, therefore one CTMC can handle a maximum of 32 inputs and 32 outputs.

The AFGWC computer system has four CTMCs, two (CTMC I and II) are connected to system I and two (CTMC III and IV) are connected to system IV. All classified lines connect to CTMC II. All lines that penetrate outside the immediate AFGWC area are encrypted.

SECURITY

AFGWC has a requirement to operate two of its computers (Systems II and III), with Top Secret, SI and SAO, and two of its systems (I and IV) in a mixed mode with classifications up to Top Secret. There is a future possibility of needing to handle SIOP ESI information on the AFGWC machines.

FUTURE TRENDS

It is expected that more interaction with other systems will be required in the future. A near term example is that of exchanging information with SACCS instead of the current approach of only sending information.

Machines of the class of STAR, ILLIAC IV, and ASC may be used in the future and be required to be secure. This class of machines would present very difficult security problems.

SECURITY SOLUTIONS

The computer systems operate in controlled areas and are manned 24 hours a day, 7 days a week, by appropriately cleared personnel. A Security Management Organization consisting of the Senior Intelligence Officer, the System Security Manager, and System Security Officer has the responsibility to ensure proper security of the system.

Although the computer systems operate in a multiprogramming mode, all software is controlled so that a demand user can access the system, only to cause selected programs to be executed.

The following elements comprise the software security systems:

Security Control Programs;

Protected Programs;

A Security Audit Log;

A Classified Tape Log; and controls on handling and marking of classified input and output.

STRATEGIC AIR COMMAND

OVERVIEW

The Strategic Air Command (SAC) controls both a bomber force and a ballistic missile force. The control of these forces requires several data processing systems, some of which are quite extensive, and use of communication channels extending over continental distances. The SAC data processing complex has reached such a size, and become so essential for the operations of SAC, that an Assistant Chief of Staff for Data Systems has been created as a single manager by SAC. He is tasked with supporting SAC Command and Control, Intelligence, and Management Information data processing systems. He controls some 900 programmers and systems analysts and approximately 35 machines. His organization contains a Security Branch. The SAC representative on the Requirements Working Group was Maj. Walter **A.** Kujawa who is Chief of the Security Branch.

SYSTEMS

SAC has a wide variety of machines and systems. From the data processing standpoint the requirements for these systems include the full range of transaction processing, file maintenance, and general programming. SAC meets these requirements on medium to large sclae resource-sharing computer systems, provided local and remote batch as well as time-sharing services, and interfacing with both SAC-controlled and non SAC-controlled networks of computers.

The machines used include IBM 360/44, 360/50, and 360/85, Honeywell 635 and 6070, Univac 1218 and 1106, and Burroughs 3500.

The systems include an online graphics processing system, PACER, which uses a Honeywell G635 and security techniques similar to those of Advanced Logistics System. PACER is connected to the Visual Analysis Subsystem (VASS) which is an intelligence support graphics system that uses a linked IBM 360/50 and Univac 1218. Other systems of consequence at SAC are the Defense Support Program (DSP) computer (a Univac 1106) and the base level computers (Burroughs 3500s).

The SAC Automated Command Control System (SACCS) is the apex of the systems important to SAC operations. Its major function is force control. Further uses include Single Integrated Operations Plan (SIOP) planning, War Gaming, and staff support. There is also a unique online requirement of the 4000th Support Group to use the SACCS computer system approximately every 15 minutes and to receive a response within 2 seconds.

SACCS consists of three major subsystems: the Data Transmission Subsystem (DTS); the Data Processing Subsystem (DPS); and the Data Display Subsystem (DDS).

The DTS connects the DPS with all the SAC command posts at about 50 different locations at the SAC bases and SAC Numbered Air Force Headquarters.

The DPS currently uses three AN/FSQ-31(V) Data Processing Centrals, three IBM 1401s, and an IBM 7090, and two IBM 1460s. All these machines are being replaced with two dual processor Honeywell 6070 WWMCCS machines, which will interface with the SAC Automated Total Information Network (SATIN) store and forward communications processor. SATIN will be interfaced, in turn, to the 4000th Support Group's terminal (which is connected via a minicomputer to the 4000th's data network), the SACCS DTS, and AUTODIN. In addition, the DPS WWMCCS machines will be directly connected to an IBM 360/44 which is connected to VASS which in turn interfaces with the Defense Support Program (DSP) computer.

The DPS passes information to the DDSs, located at Headquarters SAC and the Numbered Air Force Headquarters. The DDS supports a total of about 250 different displays of a variety of types including large screen displays.

The foregoing is not a complete summary of the SAC system configuration, but rather illustrates its complexity.

The software for the present SACCS is specially tailored to the system, allowing it to operate rapidly and to achieve efficient use of resources. SAC is concerned that the WWMCCS-supplied software for the new system may be more generous in its use of resouces and may also not be able to perform the necessary jobs in a timely fashion. The following comparison typifies that concern: whereas the current FSQ-31 operating system requires 15K 48-bit words of memory, GCOS, the WWMCCS operating the system, can require as much as 100K 36-bit words.

SECURITY

The SAC system operates in a security environment that includes all levels of classification as well as special need-to-know restrictions, e.g., Restricted Data, Special Access Required (SAR), SIOP-ESI, SI, and SAO. Ninety percent of the processing is classified; Sixty percent is Top Secret and most of that is SIOP-ESI. Present day operation treats all output as Top Secret-ESI until it's classification is established.

SAC has an operational requirements for multilevel secure systems which allow the processing of various classification levels and the support of users and terminals which are not cleared for the highest classification level being processed on the system. An example of this requirement is the fact that the 4000th support group, who are not cleared for SIOP-ESI but have a secret SAR environment, will be requiring a 2 second response time every 15 minutes from a machine that handles SIOP-ESI. A Required Operational Capability (ROC) for multilevel security is being prepared by SAC.

At present all outputs and machines are in secure vault areas. The programmers' clearance is to the highest level handled by the system.

FUTURE TRENDS

The SACCS is a significant part of the WWMCCS network, which is expected to grow in size and complexity. The Safeguard ABM, the Advanced Airborne Command Post, and the Defense Support Program will be key to this increase. The implication of the increase is that more information will be available more rapidly to SACCS, implying a broader and more current data base for command and control decisions. For example, since more options could be considered and decisions made in real-time, online crisis management is possible in which coordinated response options are compared, selected and controlled. To do that, however, will require rapid response times while the systems are handling great masses of very sensitive data. Since the resources will be pressed by these requirements, information security must be achieved by methods that will not significantly degrade response times and will minimize the use of resources.

SECURITY SOLUTIONS

SAC has formulate a phased security plan. In Phase I of the plan, unclassified processing will be done and there will be a minimal implementation of user catalogs, permissions, and passwords. SAC will perform TEMPEST tests of the system installations and make comparisons of the test results with those of previous Electromagnetic Interference (EMI) tests.

Phase II of the plan begins with the development of purge procedures capable of meeting AFR-205-25 (which defines the regulations for safeguarding the SIOP). These will include procedures for purging the WWMCCS communications processors, the CRT buffer areas, and the disks. Next SAC will develop a classified test data base with all of its processing and output contained in a secure vault area. The test data base will use a single machine to handle all classifications from Top Secret down through Confidential. Additionally, the testing during this period will address system integrity; i.e., the determination of the extent to which GCOS can reliably handle multiprogramming. This effort will require a minimal password capability and SAC will begin the development of user catalogs. In handling output, SAC will stamp all hard copy both front and back and safeguard it as though it were SIOP-ESI, until the programmers downgrade it. During this period, contractor support of the programming will pose a problem, because the contractors would require SIOP-ESI clearance.

The plans call for beginning by October of 1973 to use the new machines operationally. It was assumed that by then SAC would have completed the communications packages, the GCOS file and control procedures, and the interrupt routines, and the DoD direction to establish certification will be available. Given the required procedures and guidance, and compliance with them, the system would operate with multilevel security.

Implementation of the preceding plan will require detailed consideration of many topics such as the following:

(a) The verification of GCOS

(b) System integrity

(c) Separation of work

(d) Control of priority

(e) Catalog and file protection

(f) Permissions

(g) Passwords

(h) Remote terminals

(i) Lockouts and alarms

(j) Disallowing changes in support packages

(k) The control of output data for teletypes

(l) CRTs and hard copy from CRTs

(m) Accountability and audit trails

(n) Security packages for the WWMCCS communications processor

(o) The Data-Net 355

(p) System restoration and recovery

(q) Development of tests

(r) Purges, recertifications, and lockouts

(s) Security routines for SATIN and the 4000th Support Group

(t) Interface security packages for SATIN and SAC DMS

(u) System verification including initial certification, recertification, and an online testing program

(v) Continuously testing the hardware and software with the user's programming

(w) Hardware and software subverter

For day-to-day operations, a security "command post"-like activity will make decisions on such problems as safe data and recovery. This should be an online operational capability.

MILITARY AIRLIFT COMMAND (MAC)

OVERVIEW

The Military Airlift Command runs one of the world's largest airline operations, handling both passengers and cargo. MAC has the responsibility to airlift troops and equipment for all the armed forces. MAC has data processing equipment that supports its automated cargo and passenger handling systems. These systems are currently being updated. In addition to this updating, an expansion of system functions to include management information is in the process of development. This new integrated system is termed the MAC Integrated Management System (MACIMS). The MAC representative on the Requirements Working Group was Lt. Col. Jack Reed.

SYSTEM

The MAC Integrated Managements System will feature a fully integrated data base, and will satisfy the online processing requirements of MAC's day-to-day business as well as those of command management. MACIMS will primarily use WWMCCS ADP program hardware. In addition to their MACIMS functions, the WWMCCS machines will be used in support of the National Command Authority. MACIMS is scheduled to become operational about 1975, and to have a five year life.

MACIMS will use a complex of three WWMCCS computers to be collocated at Scott AFB, Illinois. 2400 bps lines will mutually interconnect the three computers and 70 or more remote interactive terminals will interface with each of them. The computers will also be accessed, via AUTODIN, by terminals at overseas locations. The majority of the terminals will do only unclassified work, but about a third of them will be eligible to receive classified information. MACIMS will interconnect with the Air Weather Services' forecasting system and will use data from that and other systems to do computer flight planning.

SECURITY

MACIMS is required to process both classified and unclassified data, and to serve both and non-secure online terminals, simultaneously. The work load will be about 97% unclassified.

MACIMS will also have an aggregation problem. Large amounts of unclassified data are being brought together into a structured data base, from which it will be accessible and can be used to assemble classified information.

MAC has stated a requirement that any authorized user should be able to use any MACIMS terminal to do anything in the system. This requirement precludes the applicability of those security approaches that restrict access to classified data to only specific terminals which have a validated security clearance. The requirement

instead demands more sophisticated approaches. Another problem arises from the fact that important uncertainties exist concerning the status of the WWMCCS security system.

FUTURE TRENDS

Currently most of the MAC system is running unclassified. Most of the classified work relates to doing things in support of other commands and is classified up to Top Secret. This classified work is currently handled in specifically controlled areas. In the future MACIMS era a transition will take place from the current highly dispersed data base with its many unclassified jobs to a large integrated and structured data base, classified due to its size and scope. The new mode of operation will include unclassified users accessing what is effectively a classified data base. The data base will be effectively classified not only because of the aggregation of structured data but also because it will contain classified items of information.

There will also be increased connection between MACIMS and other systems such as the Global Weather Center and the Advanced Logistics System.

SECURITY SOLUTIONS

MACIMS will use the WWMCCS security package.

Initially, the MACIMS security approach will be that of scheduling certain periods of time for classified operation. During these periods certain portions of the system will be completely isolated from the balance of the system to allow the isolated portions to operate as a closed secure system with classified data. During these periods, however, the overall system performance will be degraded. Consequently this approach to security cannot be tolerated for long periods.

There will be password protection for the accessing of data in this interim system, but this protection will apply mostly to need to know.

ELECTROMAGNETIC COMPATIBILITY ANALYSIS CENTER

OVERVIEW

The Electromagnetic Compatibility Analysis Center (ECAC) is funded by the Air Force and is Detachment Eight under AFSC/ESD. Functionally, ECAC is under JCS Control, so that it is a DOD Component, administered by the Air Force. ECAC is a contractor operated center with 300 contractor personnel and 35 to 40 military and civilian personnel including representatives from all the services. Using mathematical models ECAC analyzes systems which contain electromagnetic generating and sensing equipment to determine if the equipments can exist together without mutual interference. ECAC does frequency management and maintains the records keeping system for the Joint Frequency Panel. They also provide data base outputs from their equipment-in-place file, and do radar siting analyses. The representative of ECAC on the Requirements Working Group was Mr. Dick Greatorex.

SYSTEM

ECAC uses a Univac 1108 with 100 million words of secondary storage and a data base of 65 to 70 million words. Currently this data base consists of the inventory of all military electronics and communications equipment in place within the CONUS and much of that outside the CONUS. File size limitations have dictated that separate files be established to accommodate the several categories of characteristics.

The Nominal Characteristics File (NCF) contains the kind of nominal technical characteristics typically included in Technical Orders for the communications equipments which are in place. The in-place equipment environment file (E FILE) contains the information that relates each equipment to its location and environment. There is a large digitized topographic file that describes many areas both in the CONUS and elsewhere. This file is especially useful for radar siting analysis, and for factoring topography into mutual interference calculations. CRT terminals are used both for maintaining and working with the data base.

The Illinois Institute of Technology Research Institute (IITRI) provides the on-site contractor personnel who do the bulk of the technical work for the facility. IITRI has generated and maintains the software, and performs most of the analysis.

SECURITY

Both the NCF and E files are Secret, however, the individual records in the file vary in classification from Confidential through Secret, and some records also contain Restricted Data. Each record in the file contains its own classification. The entire computer complex, including the CRT terminals which access it, is contained in a closed secure environment. The communication lines are not encrypted but are protected by being enclosed in the secure environment and inspected daily according to the provisions specified by the industrial security manual.

All personnel in the building are cleared through Secret, and have need-to-know for all the information in the data base, and have online access to all the data in the system. The ECAC system is thus operated as a closed single-level (Secret) system.

ECAC also has some requirements to service non-DOD agencies, notably the Federal Aviation Agency and the Federal Communications Commission, which will involve allowing them access to the data base. These requirements would be for a remote batch type of operation in which the non-DOD customer could trigger a program to access the data base and retrieve information from it. In the FAA case, all the inputs would be unclassified, but the data base would contain data classified up to Secret and all of the data requested would contain some classified information. Similarly, the FCC would like to use the ECAC data base in a secure interactive mode. Such uses would require a low-cost, secure, interactive terminal. NSA would also like to have a remote terminal from which to query the ECAC data base to obtain information to compare against NSA data bases.

FUTURE TRENDS

ECAC anticipates:

(a) an increase in the number of system users;

(b) a need for secure remote terminals which may be used by uncleared, as well as cleared users;

(c) an expansion of the system data base to include more data on such items as status of frequency assignments; and

(d) eventual tie-in to AUTODIN.

The ECAC system is currently operating as a closed system, in which everyone who has access is cleared to the Secret level and has access to all the information in the system. The lines to the CRT terminals are out in the open but inspected daily for evidence of taps or damage indicating possible compromise. No audit trail is kept on the requests, but the high water mark of the security level of data requests is recorded.

The second part of ECAC responsibilities is frequency management. To perform this function, ECAC is developing a frequency management data base which will include data obtained from field users. In order to encourage the users to give such data, ECAC recognizes that they should provide services to the users, such as allowing the users to make online queries of the data base.

SECURITY SOLUTIONS

At the time of the Working Group meeting ECAC gave no further information about security solutions. The current system is a closed system in which all personnel with access to the data are cleared, but solutions will be needed if outside agencies are to be given remote access to the ECAC data base.

TACTICAL INFORMATION PROCESSING AND INTERPRETATION SYSTEM

OVERVIEW

The Tactical Information Processing and Interpretation System (TIPI) is a modularized family of equipments designed to satisfy the complete spectrum of tactical intelligence requirements for the Air Force and the Marine Corps general purpose forces. It consists of completely militarized Automatic Data Processing Equipment. The representative to the Requirements Working Group from the TIPI system office at ESD (TYI) was Mr. M.L. Mleziva.

SYSTEM

Mr. Mleziva gave information on three principal elements of the system:

(a) the Display Control, Storage and Retrieval Element (DC/SR);

(b) Image Interpretation (II); and

(c) Image Processing (IP)

There is a two-way flow of information with the Tactical Air Control System (TACS), accommodating both queries and data.

One DC/SR would be deployed for a limited war situation, and would be located at the Numbered Air Force Headquarters (NAFH). The DC/SR will have online terminals at the Tactical Air Control Center (TACC) as well as the NAFH, and the lowest forward man at the Forward Command Post (FCP) would have access to the DC/SR data base.

The DC/SR contains an AN/UYK-7 computer which interfaces with a communications processor, a data base containing potentially 300 million characters, and AN/UYK-12 minicomputers which drive the CRT display terminals. The communication processor handles 1200-2400 bps data links and 75 bps teletype lines. A secure voice backup is provided for these channels. There is one shelter for the UYK-7, one for the disk/drum storage, and one for each CRT cluster.

The IP has 40 shelters and contains no computer.

The II contains a medium scale Texas Instruments Computer and 12 shelters. This latter element processes photos and enters information into storage.

The DC/SR is a dedicated system with no programming done in the field. The UYK-7 uses JOVIAL but the other machines in the complex use assembler level programming. The UYK-7 is a 32 bit machine in which it is relatively difficult to enter a program. There is only minimal on-site capability for patches to the system. The data base has partially inverted files.

SECURITY

The system is basically a closed system. The DC/SR is a Top Secret environment and its communications are encrypted. It appends the high watermark classification to the data processed. There are requirements for file system access to be checked. Although desirable, neither provisions nor procedures for core purging or emergency destruction of data have yet been implemented.

FUTURE TRENDS

No information on future plans or trends for this system were presented.

SECURITY SOLUTIONS

The system is a closed system. The equipment is in a protected, secured environment with cryptographic protection for communications.

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* JAMES P. ANDERSON COMPANY BOX 42 FORT WASHINGTON, PENNA. 19034	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP

3. REPORT TITLE

COMPUTER SECURITY TECHNOLOGY PLANNING STUDY – VOL II

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
FINAL REPORT FEB. – SEPT. 1972

5. AUTHOR(S) *(First name, middle initial, last name)*

JAMES P. ANDERSON

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
OCTOBER, 1972	131	4

8a. CONTRACT OR GRANT NO. F19628-72-C-0198 b. PROJECT NO. 6917 c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* ESD-TR-73-51 VOL II

10. DISTRIBUTION STATEMENT

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY ELECTRONIC SYSTEMS DIVISION, AFSC L. G. HANSCOM FIELD BEDFORD, MASS. 01730

13. ABSTRACT

DETAILS OF A PLANNING STUDY FOR USAF COMPUTER SECURITY REQUIREMENTS ARE PRESENTED. A DEVELOPMENT PROGRAM TO OBTAIN A OPEN-USE MULTI-LEVEL SECURE COMPUTING CAPABILITY IS DESCRIBED. PLANS ARE ALSO PRESENTED FOR THE RELATED DEVELOPMENTS OF COMMUNICATIONS SECURITY PRODUCTS AND THE INTERIM SOLUTION TO PRESENT SECURE COMPUTING PROBLEMS. FINALLY A RESEARCH PLAN COMPLEMENTARY THE RECOMMENDED DEVELOPMENT PLAN IS ALSO INCLUDED.

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
COMPUTER SECURITY RESEARCH AND DEVELOPMENT SECURITY MODELS OPERATING SYSTEMS COMPUTER ARCHITECTURE PROTECTION PRIVACY DATA MANAGEMENT SYSTEMS HIGHER ORDER LANGUAGES						

DD FORM 1 NOV 65 1473 (BACK)
(PAGE 2)

UNCLASSIFIED
Security Classification